VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra elektroniky

Vektorové řízení synchronního motoru s budícím vinutím s využitím moderního DSP

Synchronnous Motor Vector Control with Modern DSP

2011 Bc.Martin Soukup

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra elektroniky

Zadání diplomové práce

Student: **Bc. Martin Soukup**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2612T015 Elektronika

Téma: Vektorové řízení synchronního motoru s budicím vinutím s využitím moderního DSP
Synchronnous Motor Vector Control with Modern DSP

Zásady pro vypracování:

1. Proveďte rozbor matematického modelu synchronního motoru s budicím vinutím s ohledem na metody jeho řízení.
2. Navrhněte regulační strukturu vektorového řízení synchronního motoru s budicím vinutím.
3. Vytvořte program navržené struktury pro mikropočítačový systém se signálovým procesorem.
4. Na reálném pohonu se synchronním motorem ověřte funkci vektorového řízení a změřte průběhy nejdůležitějších veličin.

Seznam doporučené odborné literatury:

Dle pokynů vedoucího závěrečné práce

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Petr Palacký, Ph.D.**

Datum zadání: 19.11.2010

Datum odevzdání: 06.05.2011

doc. Ing. Petr Palacký, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě, 6.května 2011 ……

Martin Soukup

Na tomto místě bych rád poděkoval vedoucímu diplomové práce Doc. Ing. Petr Palacký, Ph.D. za cenné rady a doporučení. Také bych rád poděkoval, za rady a pomoc při řešení problému ing. Martinu Sobkovy.

Abstrakt

Cílem této práce je návrh a realizace struktury řídicího systému pro vektorovou regulaci v orientovaném systému souřadnic synchronního motoru s budícím vinutím za využití moderních prostředků pro realizaci regulace střídavých synchronních strojů.

Návrh tohoto řešení vychází a aktualizuje již vytvořené řešení, které nahrazuje modernějším řídicím systémem a uživatelsky přívětivějším rozhraním. Toto řešení na základě rozboru matematického modelu vektorové regulace synchronního stroje s budícím vinutím bylo použito pro tvorbu výsledného algoritmu navrženého řídicího systému.

Součástí řešení této práce je návrh a realizace rozšiřující desky vývojové sady signálového procesoru firmy Texas Instruments.

Výsledná aplikace bude sloužit pro podporu výuky jako laboratorní práce. Aplikace je navržena tak aby jí bylo možno rozšířit a vylepšit.

Klíčová slova

synchronní motor, texas instruments, digitální signálové procesory

Abstract

Goal of this thesis is control system structure design and realization for vector regulation of synchronous engine with excitation windings in oriented axis system with use of modern development tools for alternating synchronous machines regulation.

Design of this solution is inspired by and upgraded existing solution. This solution replaces older solution with new control system and more user friendly user interface. This solution based on mathematical models of synchronous engine vector regulation with excitation windings was used to create final algorithm of designed and realized control system.

Part of this solution is design and realization of expansion board for Texas Instrument´s digital signal processor development kit.

Final application will serve as teaching support material in a form of laboratory exercise. Application is designed for further extensions and improvements.

Key words

synchronous motor, texas instruments, digital signal processor

Seznam použitých symbolů a zkratek:

DSP	**Digitální signálový procesor**
eQEP	**enahnced Quadrature Encoder Pulse**
AD	**Analogově digitální**
DA	**Digitálně analogový**
SPI	**Seriál Peripheral Interface**
U-f	**Řízení napětí - frekvence**
CPU	**Central Processing Unit**

Obsah:

1 Úvod

Hlavní náplní této práce je návrh a realizace řídicího systému pro vektorovou regulaci v orientovaném systému souřadnic synchronního motoru s budícím vinutím.

Druhá kapitola obsahuje matematický popis vektorové regulace synchronního motoru s budícím vinutím, který je napájen cyklokonvertorem. Tato kapitola slouží jako teoretický úvod a podklad pro realizaci celého systému.

Třetí kapitola pojednává obecně o signálových procesorech, popisuje jejich funkce, možnosti, strukturu, architekturu, stručně historii. V této kapitole je proveden také průzkum trhu v oblasti signálových procesorů vhodných pro řešení této práce. Nastiňuje možnosti dalšího vývoje signálových procesorů. Nakonec opodstatňuje volbu procesoru zvoleného pro tuto práci, kde je určen pro realizaci řídicího systému.

Čtvrtá kapitola popisuje jednotlivé části návrhu a realizace celého systému. Popisuje využité části procesoru zvoleného v třetí kapitole. Dále popisuje cyklokonvertor, který napájí statorové vinutí synchronního motoru. Z části software je zde popsáno uživatelské rozhraní realizováno ve vývojovém prostředí labview10 a software procesoru z třetí kapitoly realizovaný ve vývojovém prostředí code composer. Poslední část této kapitoly zobrazuje a popisuje navrženou a realizovanou desku, která uzpůsobuje využití pro aplikaci řízení synchronního motoru a doplňuje o nezbytné obvody, které nejsou součástí vývojové sady.

V páté kapitole jsou zobrazeny a popsány naměřené průběhy z již realizované a zapojené aplikace synchronního motoru.

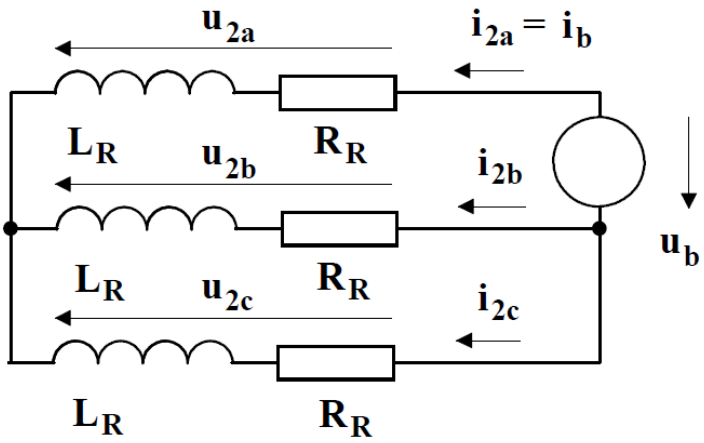
2 Popis vektorové regulace synchronního motoru

Vektorová regulace umožňuje optimalizovat chod motoru v ustálených i v přechodových jevech. Základní podmínkou vektorového řízení střídavého motoru je oddělení regulačních obvodů pro moment a magnetizační tok tak, aby se vzájemně neovlivňovali. Regulačním obvodem momentu se nastavuje moment motoru a tedy činný výkon. Regulačním obvodem magnetického toku se realizuje výsledný magnetický tok stroje a tedy jalový výkon.

Princip vektorového řízení spočívá v rozložení prostorového vektoru statorového proudu do dvou kolmých složek v rotujícím souřadnicovém systému (systém orientovaných souřadnic x, y), který může být orientován na prostorový vektor statorového magnetického toku nebo na prostorový vektor výsledného rotorového magnetického toku nebo na prostorový vektor výsledného magnetického toku.

Momentotvorná složka vektoru statorového proudu určuje spolu s příslušným vektorem magnetického toku, moment stroje. Magnetizační složka leží ve směru vektoru magnetického toku, ovlivňuje magnetizaci motoru. [1]

2.1 Matematický model synchronního motoru s budícím vinutím



Obrázek 1 Náhradní schéma symetrického rotorového vinutí napájeného se SS napěťového zdroje [1]

Pro použití modelu obecného střídavého motoru definujeme prostorový vektor rotorového napětí[1].

$$u_2^R = \frac{2}{3}\left(u_{2a} + u_{2b}e^{j\frac{2\pi}{3}} + u_{2c}e^{j\frac{4\pi}{3}}\right)$$

Po úpravě rovnice () získáme následující vztah, který popisuje budící napětí synchronního motoru [1]:

$$u_2^R = \frac{2}{3}u_b$$

Soustava diferenciálních rovnic popisující dvou pólový synchronní stroj[1]:

$$R_S * i_1^S + L_S \frac{di_1^S}{dt} + L_h \frac{d}{dt}\left(i_2^R e^{j\varepsilon}\right) = u_1^S$$

$$R_R * i_1^S + L_R \frac{di_1^R}{dt} + L_h \frac{d}{dt} * \left(i_2^S * e^{-j\varepsilon}\right) = \frac{2}{3} u_b$$

$$J \frac{d\omega}{dt} = \frac{3}{2} L_h * Im * \left[i_1^S * \left(i_1^R * e^{j\varepsilon}\right)^*\right] - m_z$$

$$\omega_m = \frac{d\varepsilon}{dt}$$

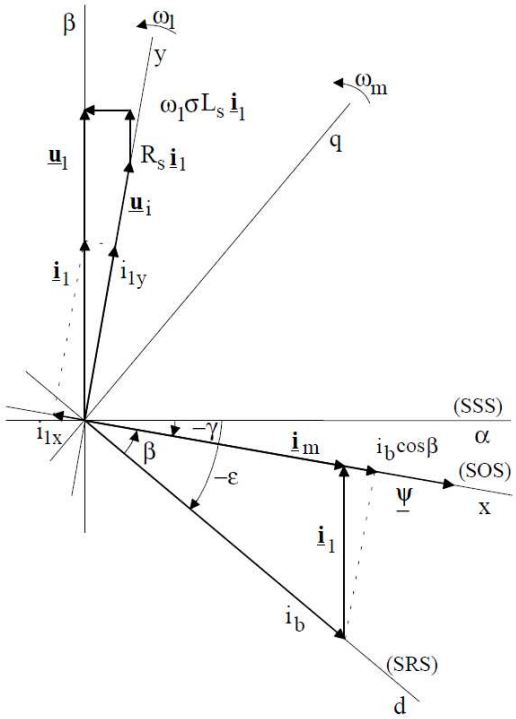
Vektorové natočení ze statorových do orientovaných souřadnic[1]:

$$x^o = x^s e^{-j\gamma}$$
$$x_x = x_\alpha cos\gamma + x_\beta sin\gamma$$
$$x_y = -x_\alpha sin\gamma + x_\beta cos\gamma$$

Výsledná Soustava diferenciálních rovnic po natočení do orientovaných souřadnic[1]:

$$\sigma T_s * \frac{di_{1x}}{dt} + i_{1x} = \frac{u_{1x}}{R_s} + \omega_1 \sigma T_s i_{1y} - (1-\sigma) T_s \frac{d_{im}}{dt}$$
$$\sigma T_s * \frac{di_{1y}}{dt} + i_{1y} = \frac{u_{1y}}{R_s} + \omega_1 \sigma T_s i_{1x} - (1-\sigma)\omega_1 T_s \; i_m$$
$$T_R \frac{d_{im}}{dt} + i_m = i_{1x} + (1 + \sigma_R) \frac{u_b}{R_b} \cos(\gamma - \varepsilon)$$
$$\omega_1 - \omega_m = \frac{1}{i_m T_R} \left[i_{1y} - \frac{(1+\sigma_R)\, u_b}{R_b} * \sin(\gamma - \varepsilon) \right]$$
$$J \frac{d_{\omega m}}{dt} = \frac{3}{2} * \frac{L_h}{(1+\sigma_R)} * i_m \, i_y - m_z = \frac{3}{2} \psi \, i_y - m_z$$
$$\omega_m = \frac{d\varepsilon}{dt}$$
$$\omega_1 = \frac{d\gamma}{dt}$$
$$T_S = \frac{L_S}{R_S}$$
$$T_R = \frac{L_R}{R_R} = \frac{L_b}{R_b}$$

Grafické znázornění vazeb mezi jednotlivými prostorovými vektory



Obrázek 2:Vektorový diagram synchronního motoru s budicím vinutim[1]

3 Popis DSP

Pod pojmem signálový procesor rozumíme mikroprocesor nebo mikroprocesorový systém, který je přizpůsoben pro rychlé zpracování signálu v reálném čase. Typickými algoritmy, které jsou vhodné pro realizaci v signálovém procesoru, jsou algoritmy číslicových filtrů (FIR – Finite Impulse Response, IIR – Infinite Impulse Response), algoritmus rychlé Fourierovy transformace, nebo pro velmi rychlý výpočet matematických modelů soustav (například střídavých elektrických strojů), který se využívá v této práci.

Většina velkých výrobců signálový procesorů jako jsou Texas Instruments, Motorola (Freescale), Analog Device, má ve svém sortimentu signálové procesory s mnoha dalšími periferiemi jako např. PWM jednotka, USB, UART, AD, FLASH, SPI, CAN, I2C, PIXEL COPROCESSOR a nespočet dalších. Tyto procesory se svými periferiemi blíží spíše mikrokontrolérům, avšak na druhou stranu svým výpočetním výkonem spíše signálovým procesorům.

Typický digitální signálový procesor je vystavěn na harvardské architektuře. Tato architektura má oproti von Neumannovu modelu počítače oddělenou paměť pro program od paměti pro data. V praxi to znamená, že data a kód programu využívají vlastní sběrnice, což zvyšuje propustnost systému.

Dalšího zrychlení výpočtů se dosahuje pomocí specializovaných výpočetních jednotek procesoru, které dokážou pracovat paralelně. Typický DSP má kromě aritmeticko-logické jednotky (ALU) navíc rychlou násobičku, která dokáže operaci násobení s přičítáním $A \leftarrow A + B.k$. Tato operace je základní operací většiny algoritmů digitálního zpracování signálu. DSP zpravidla obsahuje dvě nebo více nezávislých adresních jednotek, tzv. DAG (Data Address Generator), adresujících data v lineárních nebo kruhových bufferech. Typický DSP tak umožňuje během jednoho taktu provést jeden krok skalárního násobení dvou vektorů (vynásobení hodnot ze dvou bufferů, přičtení do akumulátoru, posun na další index v bufferech). Procesor s klasickou architekturou by na stejnou operaci potřeboval několik taktů (např. 1. načtení hodnoty z prvního bufferu, 2. vynásobení hodnotou z druhého bufferu, 3. přičtení výsledku do akumulátoru, 4. posun adresy prvního bufferu, 5. posun adresy druhého bufferu). [7], [6], [9]

3.1 Architektura signálových procesorů a jejich základní vlastnosti

Základní architektury mikroprocesorových obvodů byly postupně převzaty z návrhu číslicových počítačů. Požadavky na výpočetní výkon pak vedly nejprve k modifikacím stávajících řešení a pak dále k novým myšlenkám v uspořádání jednotlivých složek architektury mikroprocesorů. Pod pojmem architektura mikroprocesorů (stejně tak počítačů) máme, namysli obor studující **strukturu, organizaci, realizaci a funkci** mikroprocesorů. Struktura popisuje propojení jednotlivých funkčních bloků, organizace řeší dynamické vazby mezi těmito bloky, realizace se zabývá realizací vnitřních obvodů a typem použitých součástek a funkce uvádí, chování mikroprocesoru z vnějšího pohledu (vnější popis).

3.1.1 Dělení architektur mikroprocesorových obvodů

Architektury mikroprocesorových počítačů lze od jejich počátku rozdělit takto:

- von Neumannova
- **harvardská**
- speciální harvardská architektura
- LIW (**L**ong **I**nstruction **W**ord),VLIW (**V**ery **L**ong **I**nstruction **W**ord)
- superskalární
- architektura paralelních systémů

3.1.1.1

Von Neumannova architektura popisuje počítač se společnou pamětí pro instrukce i data. To znamená, že **zpracování je sekvenční** oproti například Harvardské architektuře, která je typickým představitelem **paralelního zpracování**.

Procesor počítače se skládá z řídící a výkonné (aritmeticko-logické) jednotky. Řídící jednotka zpracovává jednotlivé instrukce uložené v paměti, přičemž jejich vlastní provádění nad daty má na starosti aritmeticko-logická jednotka. Vstup a výstup dat zajišťují vstupní a výstupní jednotky.

3.1.1.2 Harvardská architektura

Harvardská architektura je p architektura, která má oddělenou programovou datovou paměť. Název pochází z počítače Harvard MarkI, který byl postaven na této architektuře.

U harvardské architektury není třeba mít paměť stejných parametrů pro data a pro program. Paměti můžou být naprosto odlišné, mohou mít různou délku slova, časování, technologii a způsob adresování. Dvojí paměť umožňuje paralelní přístup k oběma pamětem, což zvyšuje rychlost zpracování.

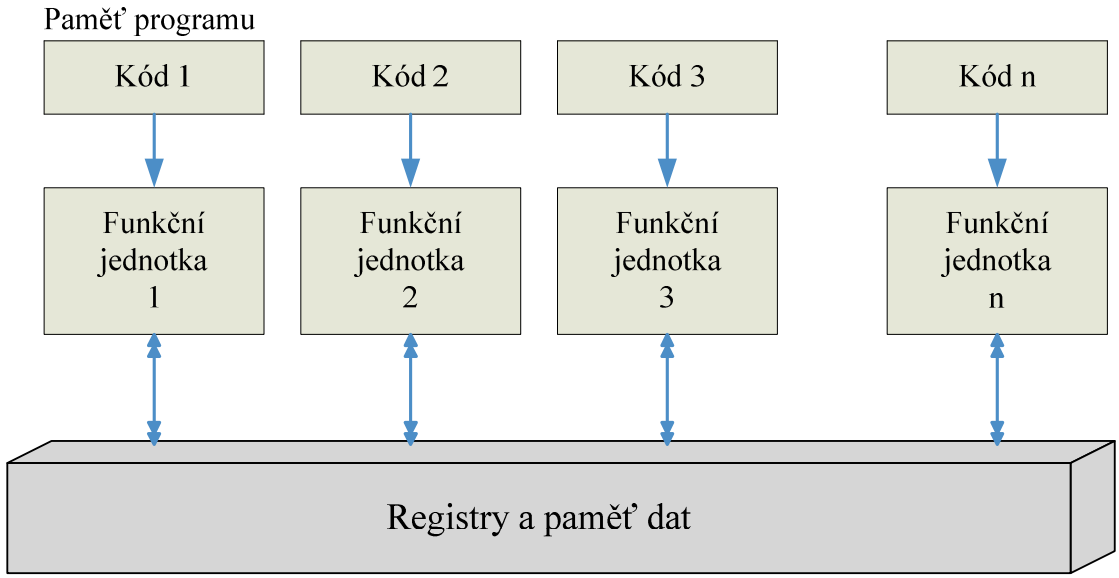
3.1.1.3 Architektura LIW a VLIW

Po harvardské architektuře, byla dalším krokem pro zvýšení výpočetního výkonu snaha zařadit do architektury více funkčních jednotek, které současně vykonávají více operací sdružených do jediné programové instrukce. Následkem toho bylo zvětšení instrukčního slova (od toho její název). Při dalším zvětšování tohoto instrukčního slova se pak architektura nazývá VLIW. Prvním příkladem byl procesor od firmy Texas Instruments TMS320C30, který navíc pracuje s pohyblivou řádovou čárkou. Architektura LIW má jeden programový čítač a vždy načítá pouze jednu instrukci z programové paměti, avšak tato instrukce se skládá

z dílčích instrukcí, které ovládají několik funkčních jednotek. Tento způsob má základ v horizontálním programování a byl již využit v řadě 3000 firmy Texas Instruments. Například procesor TMS320C30 má délku instrukčního slova 32b.

Pokroky v technologii VLSI (**V**ery **L**arge-**S**cale **I**ntegration) umožnily zvyšovat počet funkčních jednotek na čipu. Tak se objevili nové typy signálových procesorů s velmi dlouhou programovou instrukcí VLIW (**V**ery **L**ong **I**nstruction **W**ord). Jejich architektura je určena velkým počtem vzájemně propojených funkčních bloků jako jsou aritmeticko logické jednotky (ALU), násobičky, jednotky přístupu do paměti, jednotky pro manipulaci s bity apod.

Instrukce architektury typu VLIW se čtou po jedné, přesně jako u architektury von Neumannově počítači. Rozdíl je ovšem v tom, že každá programová instrukce (označována také instrukční paket) je podle přesně stanovených pravidel rozdělena na dílčí instrukce, v nichž jsou zapsány operační znaky, které ovládají každou funkční jednotku samostatně. Současně jsou také ovládány datové a adresové sběrnice a přenosy dat po nich. Blokové schéma architektury mikroprocesoru typu VLIW je na (Obrázek 3).



Obrázek 3 Blokové schéma architektury mikroprocesoru typu VLIW [6]

3.1.1.4 Superskalární architektura

Z výše uvedeného je zřejmé, že při programování mikroprocesoru typu LIW a VLIW musí programátor buď v jazyku symbolických adres, nebo kompilací z vyššího programovacího jazyka určit, které operace budou realizovány paralelně. Naopak u superskalární architektury je navržen speciální hardware uvnitř procesoru (Schedule unit), který automaticky sdruží operace zpracovávané paralelně do jediné instrukce. Sdružení operací závisí na tom, zdali jsou nějaké nepoužívané registry, zda jsou dostupná všechna vstupní data pro instrukci apod. Je nutné podotknout, že tento problém se řeší za chodu programu! Jinými slovy superskalární procesor (přesněji jeho hardware) bere na sebe zodpovědnost, za paralelní zpracování operací z programátora nebo kompilačního programu. To sebou nese řadu nevýhod, které si dále uvedeme. Tedy vážným nedostatkem této koncepce

je skutečnost, že programátor neví, jak dlouho se bude daný algoritmus vykonávat a co je horší, pokaždé ten čas může být jiný! Protože mikroprocesor může sdružit v druhém cyklu (při opakování nekonečné smyčky) operace jinak. Programátor může ovšem vzít v potaz nejhorší variantu seskupení operací a tedy nejdelší čas vykonání určitého segmentu, ale pak nemusí být výkon procesoru využit plně. Následkem výše uvedeného nejsme schopni daný program optimalizovat. U číslicového zpracování signálů, které jsou výpočtově náročné (např. řízení střídavých strojů) při značném omezení paměti a příkonu, je optimalizace kritickým bodem.

3.1.1.5 Architektura paralelních systémů

Pouze stručně. Hlavní myšlenkou zvyšování výpočetního výkonu je zvětšování podílu paralelního zpracování dat. Nejznámější třídění systémů zavedl M.J. Flynn 1966.

Dělení podle počtu programů řešených současně:

- **SI** (**S**ingle **I**nstruction **S**tream): Jeden řešený program.
- **MI** (**M**ultiple **I**nstruction **S**tream): Více řešených programů.

Podle počtu datových souborů zpracovávaných současně:

- **SD** (**S**ingle **D**ata **S**tream): Jeden zpracovávaný tok dat.
- **MD** (**M**ultiple **D**ata **S**tream): Více zpracovávaných toků dat.

Kombinací těchto možností vzniknou čtyři typy paralelních systémů:

- **SISD:** Typický von Neumanův počítač s jedním programem a s jedním sériově přiváděným tokem dat.
- **MISD:** Hypotetická kombinace několika programů zpracovávající jediný tok dat. Tento typ se prozatím nepoužívá a je nesprávně zaměňován se systémem s řetězeným zpracováním instrukcí.
- **SIMD:** Větší počet funkčních jednotek pracuje na řešení jediného programu. Všechny jednotky provádí tutéž operaci, ale s jinými daty.
- **MIMD:** Obecný typ paralelního systému, který obsahuje jednotky již tak samostatné, že každá z nich plní samostatný program a přitom zpracovávají jiná data.

3.2 Trend vývoje architektury signálových procesorů

Míra paralelismu kterou lze v mikroprocesoru použít je převážně závislá na úrovní na aktuální úrovni technologie výroby integrovaných obvodů. A tedy právě teď jsou aktuální trendy ve vývoji signálových procesorů následující:

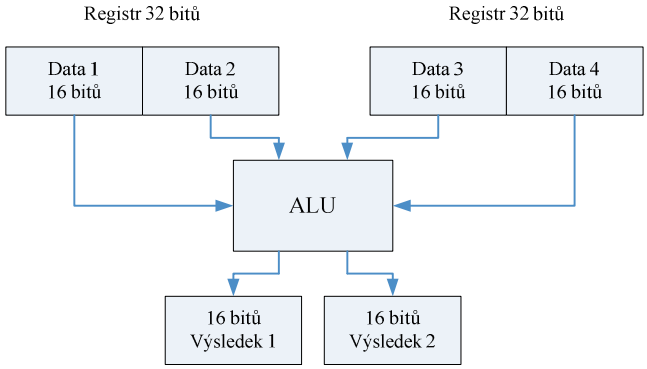
- Paralelní zpracování instrukcí
- Paralelní zpracování dat
- Využívání vyrovnávací paměti CACHE.

3.2.1 Paralelní zpracování dat:

Vícenásobné čtení a zpracování během jednoho hodinového cyklu (Multi-Isseu Prosessing) je jednou z možností, jak zvýšit stupeň paralelismu. Paralelismu lze také

dosáhnout pomocí koncepce typu **SIMD**. Tento přístup dovolí procesoru provádět stejnou operaci pomocí jedné instrukce pro skupinu nezávislých dat. Procesor na principu **SIMD** může pracovat s dlouhými registry (např. 64 bitů) jako násobky menších datových slov, pro něž provádí stejnou operaci, generuje více nezávislých výstupů.

Princip typu SIMD se stal populární v 90. letech minulého století jako prostředek pro zvýšení výkonu tehdy existujících CPU architektur pro vektorové operace, které jsou velmi často používány v multimediálních aplikacích a při zpracování signálů. Tento přístup významně zvyšuje rychlost procesorů pro algoritmy pracující s vektory, kde jsou operace přirozeně přizpůsobené pro paralelní zpracování. Právě tento princip umožnil mikroprocesorům pro všeobecné použití, jako je Intel Pentium III a Motorola PowerPC G4, konkurovat signálovým procesorům s pohyblivou řádovou čárkou co do rychlosti zpracování algoritmů číslicového zpracování signálů. Ačkoliv kořeny principu typu **SIMD** jsou u mikroprocesorů pro všeobecné použití, využití zpracování typu **SIMD** v signálových procesorech se nyní rozšířilo a zcela běžně se používá v nejnovějších čipech. Úroveň podpory pro operace typu **SIMD** se však stále dosti liší. Na příkladu signálového procesoru firmy Analog Devices si ukážeme základní myšlenku využití principu **SIMD**. Při tvorbě signálového procesoru ADSP-2116x firma modifikovala základní koncepci signálového procesoru s pohyblivou řádovou čárkou přidáním druhé skupiny výkonných jednotek, které kopírují původní návrh. Každá z obou skupin obsahuje jednotku MAC, jednotku ALU, jednotku pro posun a každá skupina disponuje vlastní operační registry. Signálový procesor může číst jednu instrukci a zpracovat ji paralelně ve dvou datových cestách používajících různá data, čímž u některých algoritmů lze dosáhnout dvojnásobného výkonu. Způsob výpočtu s využitím principu typu **SIMD** je vidět na obrázku (Obrázek 4).



Obrázek 4 Paralelní zpracování dat metodou typu SIMD [6]

Dvě a dvě vstupní 16bitová slova lze zpracovat paralelně. U algoritmů, které jsou svou podstatou zpracovatelné sériově-jako operace, které využívají výsledky jedné operace jako vstupní hodnoty druhé operace-se nedoporučuje **SIMD** využít. Programátoři musejí uspořádat data v paměti tak, aby zpracování metodou **SIMD** proběhlo plnou rychlostí. Data lze například uspřádat pamětí současně po čtyřech operandech. Často bude nutné přepsat stávající algoritmus. Tak procesor musí vykonávat dodatečné operace pro přeorganizování a čtení dat, připravených pro provedení instrukcí **SIMD** nebo sečtení všech mezivýsledků. Tyto požadavky na přeorganizování nebo přepsání algoritmu zmenšují výhody této metody. Ue signálových procesorů s pevnou řádovou čárkou principu **SIMD** využívá také řada TMS320C64xx firmy Texas Instruments.

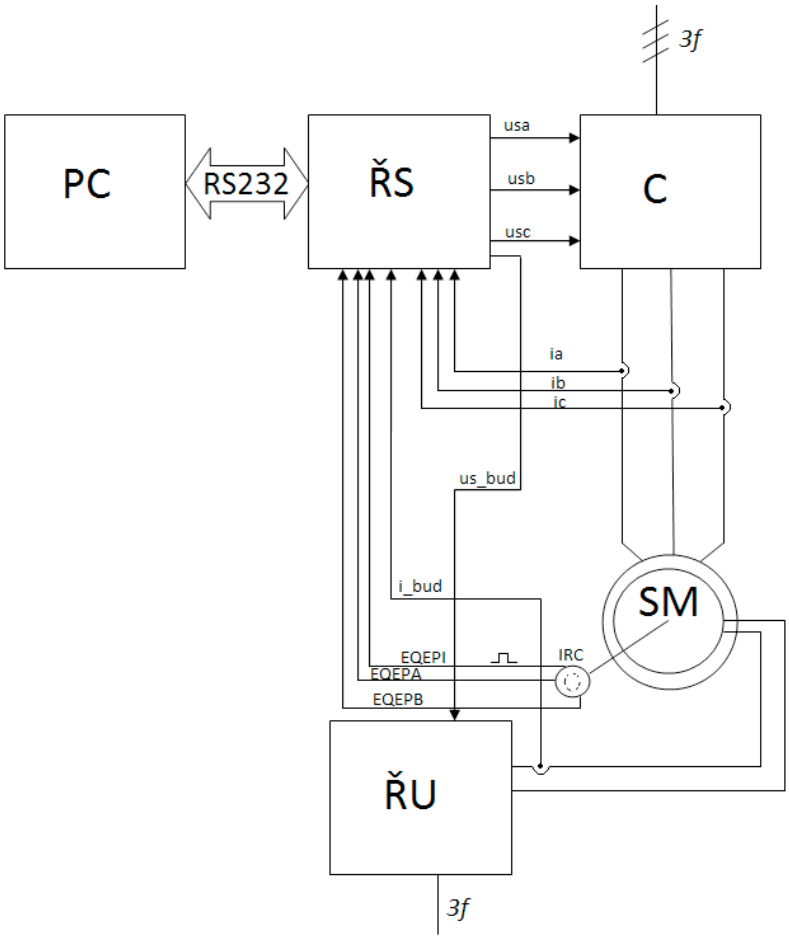
3.2.2 Využití vyrovnávací paměti cache

Velký kmitočet hodinového signálu řady mikroprocesoru vyžaduje využití extrémně rychlých pamětí k tomu, aby zpracování probíhalo plnou rychlostí. Rychlost pamětí je vykoupena jejich velikou cenou a proto se objevují snahy buď snižovat kapacity vnitřních i vnějších pamětí nebo použít vyrovnávací paměti typu cache. Jsou to skupiny rychlých pamětí s malou kapacitou umístěných blízko jádra procesoru a slouží jako prostředník mezi procesorem a pomalejší velkou vnitřní nebo vnější pamětí. Jejich hlavní přínos je v tom, že snižují požadavky na rychlost, a tudíž i cenu hlavních pamětí, která je v procesoru umístěna. Paměť typu cache je určena k uložení a čtení často používaných instrukcí anebo dat a je dynamicky plněna. Pokud vznikne požadavek na přístup do pamětí, je nejprve ověřeno, zda požadovaná data nebo instrukce jsou přístupná v paměti cache. V takovém případě čtení nebo zápis proběhne plnou rychlostí. Pokud však požadovaná data nebo instrukce v paměti cache nejsou, musí procesor čekat, než se paměť cache naplní. Časová náročnost programu se může v závislosti na obsahu této paměti. Typicky jsou paměti cache využívány u mikroprocesoru pro všeobecné použití. U signálových procesorů se vyskytují jen zřídka právě kvůli neurčitému odhadu času nutného k provedení programu. Po zjištění výhodných vlastností, které paměť cache přináší, se výrobci snaží najít způsob, jak přizpůsobit tuto paměť pro aplikace číslicového zpracování signálu. Až donedávna těch několik signálových procesorů, které používalo paměť typu cache, ji využívalo pouze pro uložení instrukcí, ale nikoliv dat. Tyto paměť pro uložení

4 Návrh a realizace

Celý systém určený k vektorové regulaci synchronního motoru s budícím vinutím, se skládá z následujících částí, která je zobrazena na (Obrázek 5):

- Synchronního motoru s budícím vinutím
- Cyklokonvertoru
- Řízeného usměrňovače
- Řídicího systému
- Osobního počítače s řídícím uživatelským rozhraním



Obrázek 5 blokové schéma systému vektorové regulace

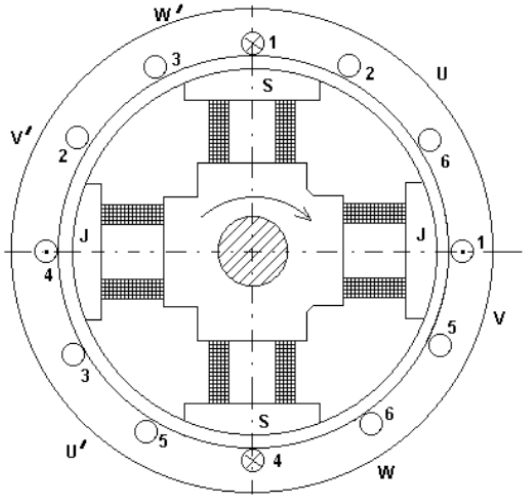
4.1 Synchronního motoru s budícím vinutím

Jedná se o stroj z vyniklými póly se dvěma pól-páry. Napájení budícího vinutí je přivedeno z řízeného usměrňovače na kroužky rotoru. Motor typu A 180M04, výrobek MEZ Frenštát má parametry uvedené v tabulce (Tabulka 1).

Tabulka 1 Parametry motoru

Parametr motoru	Hodnota
P_n	12,8kW
S	16kW
n_n	1500 ot/min
p	2
f_n	50Hz
Mn	85Nm
Jm	0,25kg*m^2
Isn	23A
cosφ	0,8

Na obrázku (Obrázek 6) je vidět příčný řez motorem.



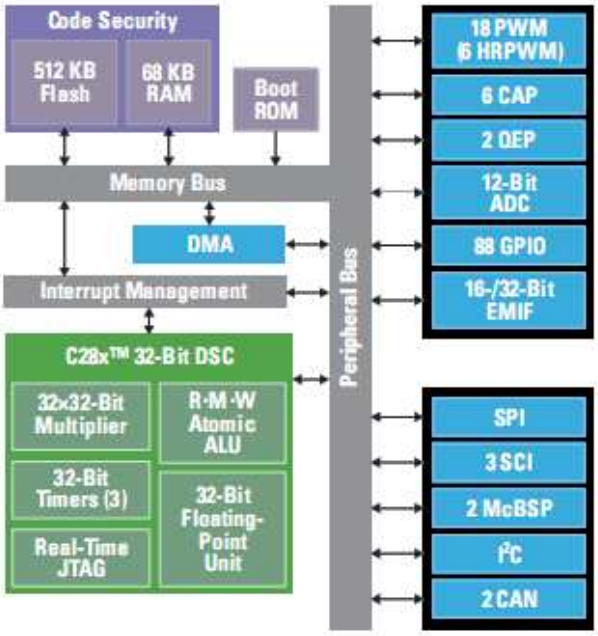
Obrázek 6 řez 4. pólovým synchronním motorem

4.2 Řídící systém

Pro realizaci vektorové regulace je nutný výkonný signálový procesor. V Našem případě byl zvolen 32-bitový signálový procesor Texas Instruments TMS28F335 sazen ve vývojovém kitu firmy Spectrum Digital eZdsp^{TM}F28335.

4.2.1 Procesor TMS320F28335

Procesor TMS320F28335 obsahuje další periferie vhodné pro zvolenu aplikaci. Tento procesor pracuje na taktu 150MHz.

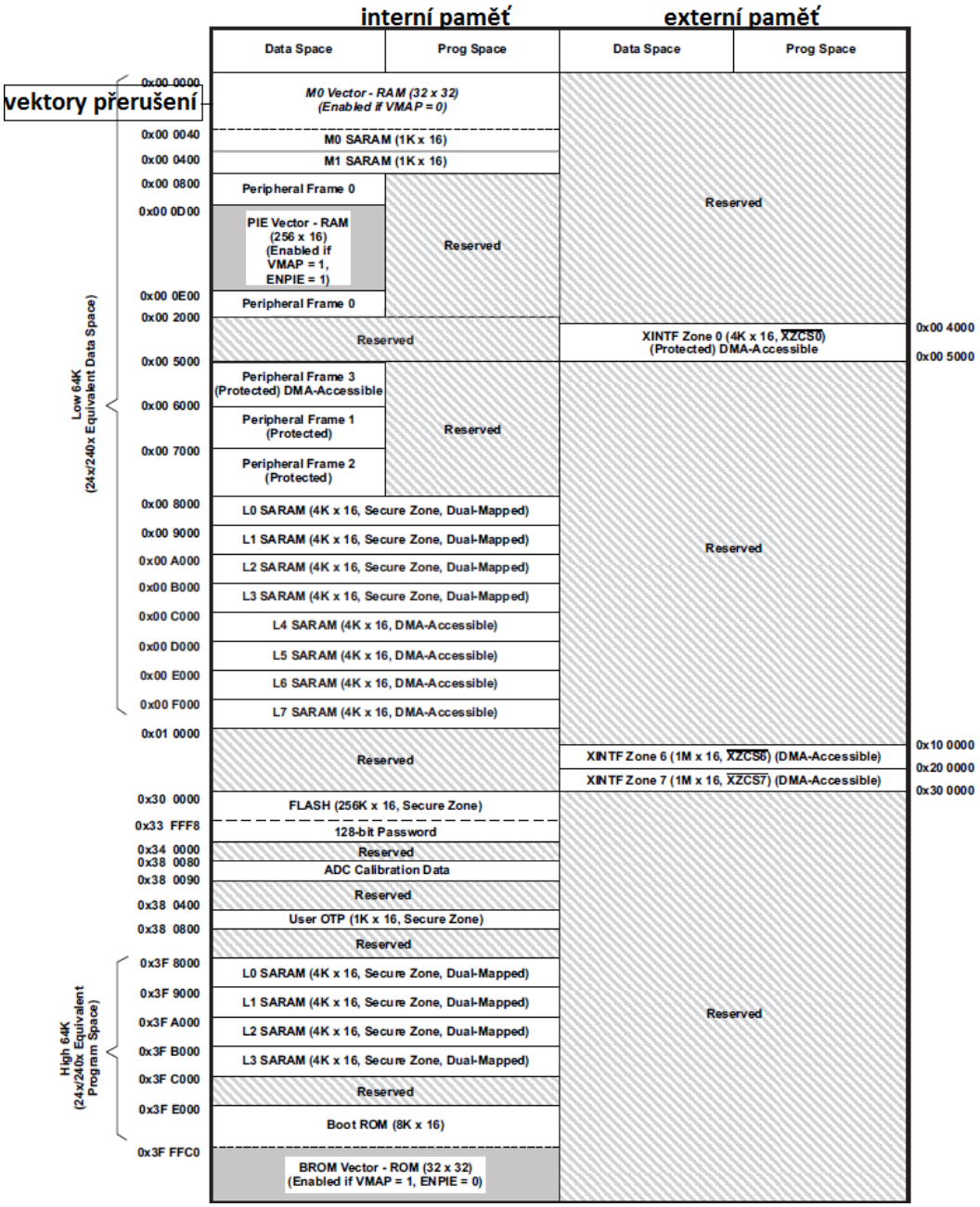


Obrázek 7: Blokové schéma procesoru TMS320F28335[5]

Periférie procesoru TMS28F335:

- 6-kanálový DMA řadič (řadič pro přímý přístup do paměti)
- **Tři 32-bitové CPU časovače – hlavní systémové přerušení pro vekt. regul…**
- 6 PWM modulů (ePWM1, ePWM2, ePWM3, ePWM4, ePWM5, ePWM6)
- **dva QEP moduly (eQEP1, eQEP2) – jednotka pro vyhodnocení polohy**
- **ADC moduly – využito pro měření výstupních signálů z proudových čidel**
- dva řadiče pro místní síť (eCAN) moduly (eCAN-A, eCAN-B)
- **tři rozhraní sériové komunikace (SPI) modul (SPI-A) – komunikace s PC a rozhraní SPI - komunikace DA převodníkem (DAC7718)**
- I2C komunikační obvod
- dva vícekanálové sériové porty, moduly (McBSP-A, McBSP-B)
- **Digitální I/O porty se sdílenými funkcemi – povolení funkce externího DA převodníku (DAC7718)**
- Externí rozhraní (XINTF)

Procesor TMS28F335 obsahuje 256kB paměťi typu Flash, 34kB paměti SARAM (Single Acces RAM), 1kB jednou programovatelné paměti (OTP) ROM, jak lze vidět na obrázku (Obrázek 8).

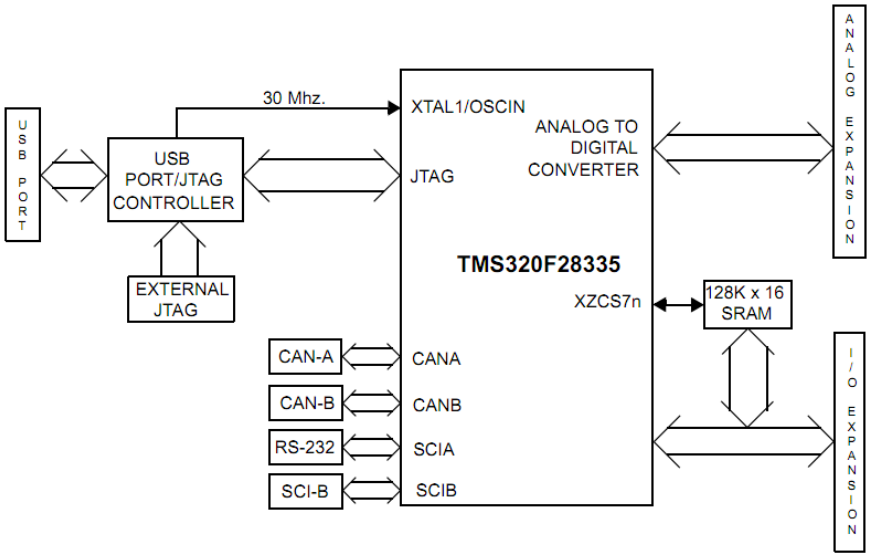


Obrázek 8 mapa paměti signálového procesoru TMS28F335[5]

4.2.2 Vývojová sada Spectrum Digital eZdsp^{TM}F28335

eZdsp^{TM}F28335 je samostatná karta, která dovoluje vývojářům použít procesor TMS320F28335 při vývoji aplikací a zjistit zda vyhovuje jejím požadavkům. Vývojová sada je také vhodná pro testování softwaru pro výše zmíněný procesor. Deska obsahuje několik rozšiřujících konektorů pro připojení testovaných periferních zařízení (viz Obrázek 9). Vývojová sada obsahuje také ovladače a emulátor pro vývojové prostředí Code Composer Studio, díky kterým je možno testovat a ladit aplikaci přímo v zařízení za běhu.
Vývojová sada obsahuje:

- TMS320F28335 DSP CPU, 150 MHz
- 32-bit FPU (Floating Point Unit)
- 68K bytů on-chip RAM v CPU
- 512K bytů paměti Flash v CPU
- 256K bytů paměti SRAM mimo CPU
- 16-vstupý 12-bit AD (Analog to Digital) převodník
- RS-232 konektor s budičem a převodníkem napěťových úrovní na desce
- CAN 2.0 rozhraní s budičem a převodníkem napěťových úrovní na desce, konektor
- Několik rozšiřujících konektorů (analogové, I/O)
- USB JTAG řadič osazený na desce
- Napájení 5V přes AC adaptér, který je součástí balení
- IEEE 1149.1 JTAG konektor na desce

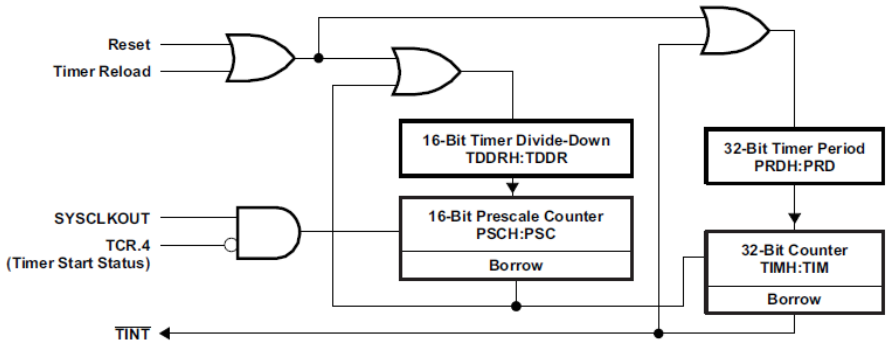


Obrázek 9 Blokové schéma vývojové sady EZDSP[5]

4.2.3 Interní časovač

Procesor obsahuje celkem 3. Interní časovače (CPU-TIMER0/1/2)

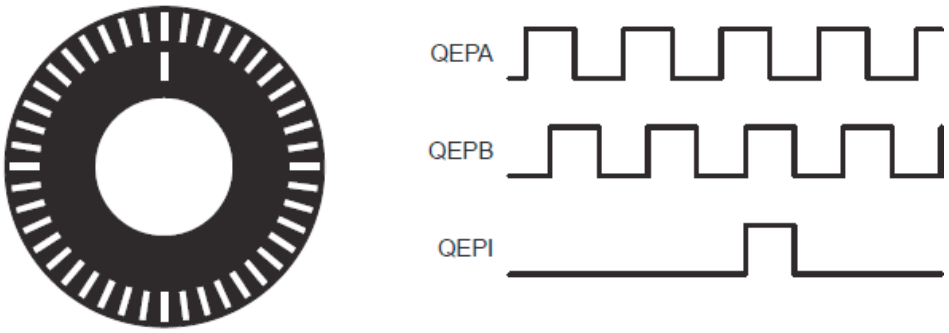
Časovač CPU-TIMER2, je obsazen a použit pro DSP/BIOS™. Časovače (CPU-TIMER0/1) jsou k dispozici pro uživatelské aplikace. V případě, že časovač CPU-TIMER2 není použit pro DSP/BIOS™ může být tento časovač použit pro uživatelskou aplikaci.



Obrázek 10 Blokové schéma vnitřního časovače[5]

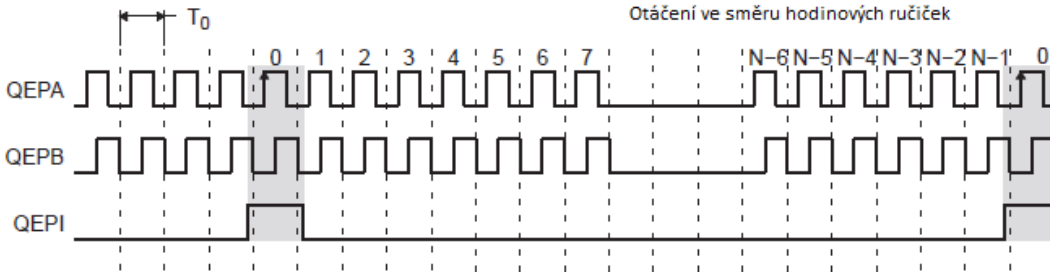
4.2.4 Interní eQEP modul

eQEP (enhanced Quadrature Encoder Pulse) modul se používá k přímému propojení s inkrementálními čidly ke zjištění pozice, směru a rychlosti z točivého stroje v pokročilých a výkonných systémech řízení, jako je například systém řízení metodou vektorové regulace použitý v této práci.



Obrázek 11 Inkrementální čidlo a signály modulu eQEP[5]

Při otáčivém pohybu dva snímací členy generují dva signály fázově posunuty o 90˚ QEPA a QEPB. Směr otáčení ve směru hodinových ručiček je dán QEPA signálem, který předchází QEPB signál, respektive naopak ve směru opačném, tak jak je vidět na průbězích na obrázku (Obrázek 12).



Obrázek 12 Průběh signálu čidla při otáčení ve směru hodinových ručiček[5]

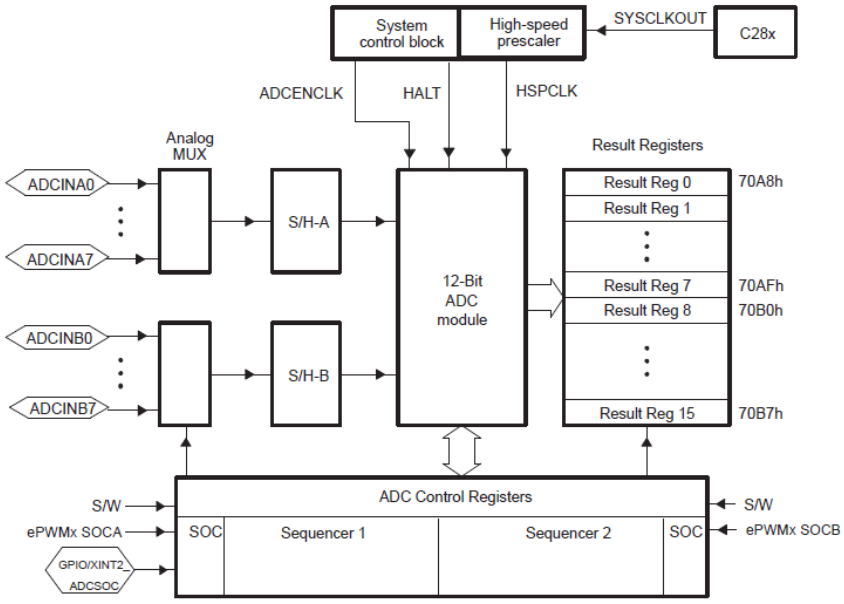
4.2.5 Interní modul AD převodníku

The TMS320x2833x ADC modul je 12-bitový analogově digitální převodník. Analogová část obvodu „jádro“ obsahuje analogové multiplexery, sample-and-hold (S/H) obvody, převodní jádro, napěťové regulátory a další podpůrné analogové obvody.

Digitální obvod „wrapper“ obsahují programovatelný sekvencér pro převod hodnot, registr s výsledky převodu, rozhraní k analogovým obvodům, rozhraní k periferní sběrnici a rozhraní k dalším modulům na čipu procesoru.

Parametry interního AD převodníku:

- 12-bitové jádro se zabudovaným dvojitím S/H obvodem
- Souběžně nebo sekvenčně vzorkovací módy
- Analogové vstupy 0V-3V
- Rychlé časy převodu na frekvenci 12,5MHz (6,25MSPS)
- 16-ti kanalálové multiplexované vstupy
- Autosekvenční schopnost poskytuje až 16 „autopřevodů“ najednou. Každý z převodů může být naprogramován k vybrání jakéhokoliv z 1-16 vstupů
- Šestnáct výsledkových registrů (individuálně adresovatelné) k uložení hodnot převodu
- Digitální hodnota analogového vstupu je odvozena dle následujících podmínek:
  Digitální hodnota = 0, když $U_{ANA} = 0$
  Digitální hodnota = $4096 \times \frac{U_{ANA}-ADCLO}{3}$, když $0 < U_{ANA} > 3V$
  Digitální hodnota = 4095, když $U_{ANA} > 3V$



Obrázek 13 Bloková struktůra interního AD převodníku[5]

AD převodník slouží v aplikaci ke čtení 4. signálů z proudových čidel LEM, kde 3 čidla měří statorové proudy a poslední čidlo měří proud procházející budícím vinutím.

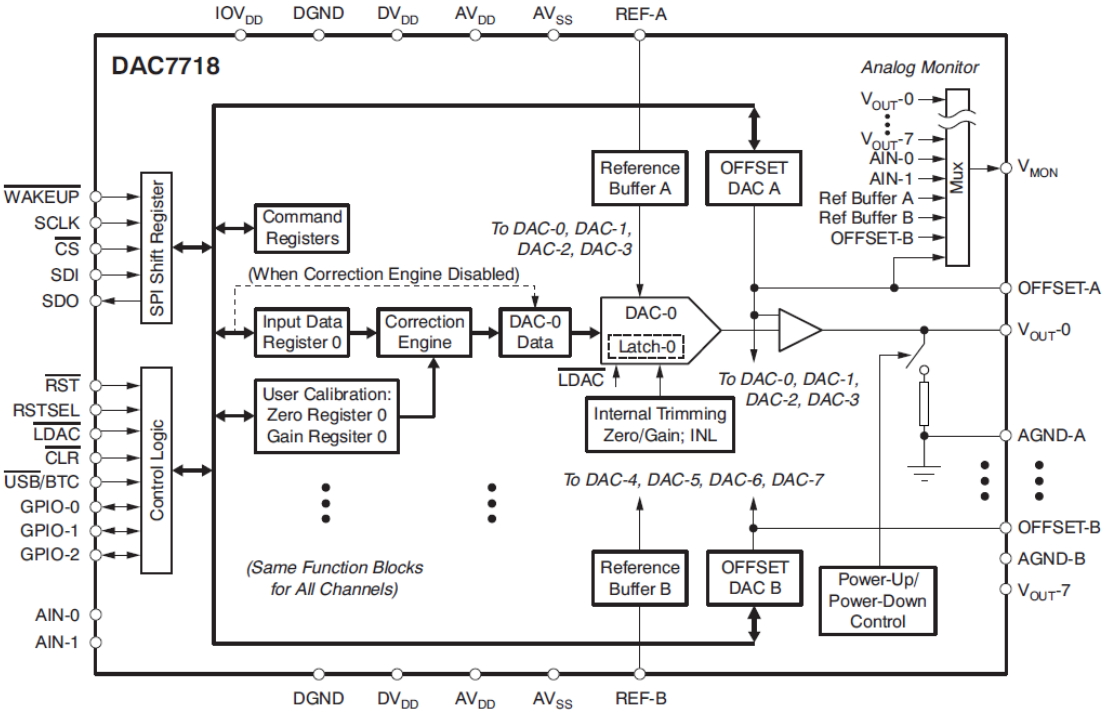
4.2.6 Komunikace s osobním počítačem a s externím DA převodníkem SPI

Pro komunikaci s osobním počítačem je použita sériová linka RS232, která je buzena obvodem MAX3238, který je řízen pomocí rozhraní SPI. Sériové linky SPI se využívá také pro komunikaci s externím DA převodníkem (4.2.7).

4.2.7 Externí DA převodník

Byl použit externí DA převodník od firmy Texas instruments DAC7718 s komunikací přes rozhraní SPI. Tento DA převodník je svými vlastnostmi vhodný pro aplikaci vektorové regulace. Pro cyklokonvertor je nutný bipolární tři řídící signály +/- 10V, které tento převodník umožňuje. Pro řízený usměrňovač je zapotřebí 1 řídící signál 0-10V, tento signál určuje úhel otevření tyristoru v řízeném usměrňovači. Parametry DA převodníku:

- 8 kanálu
- 12 bitové rozlišeni
- Volitelné logické urovně 1,8V/3V/5V
- Unipolarní výstup 0-30
- Bipolarní +/-16,5

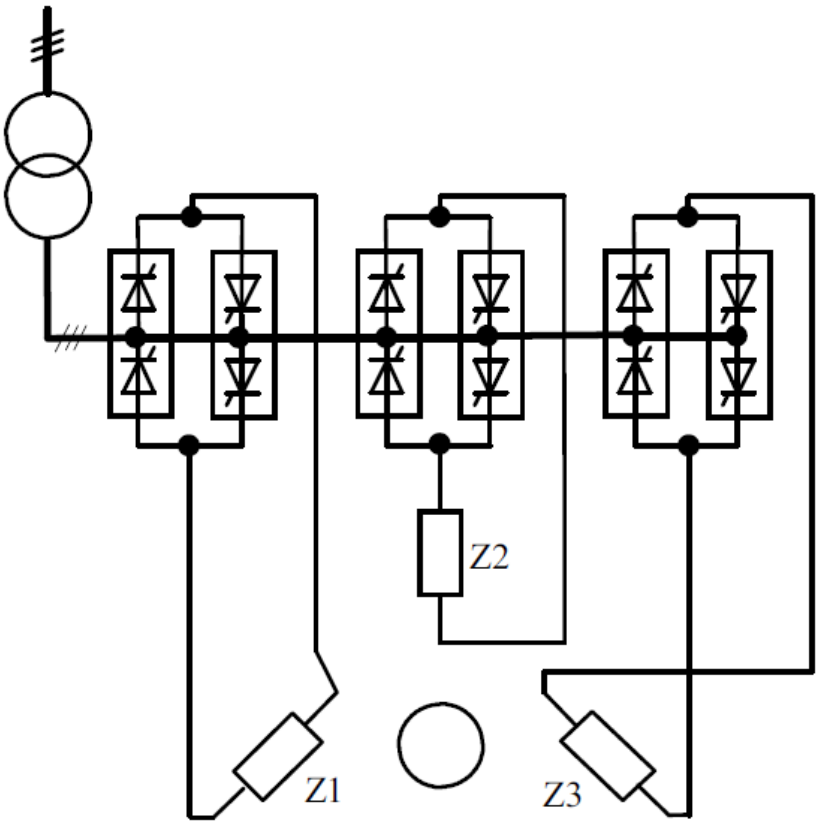


Obrázek 14 Blokové schéma DA převodníku DAC7718[5]

4.3 Cyklokonvertor

Cyklokonvertor patří do skupiny přímých měničů kmitočtu, kdy se průběh výstupního napětí vytváří přímo z části křivek trojfázového napětí o kmitočtu 50Hz. Vhodným řízením čtyř kvadrantového řízeného usměrňovače (Obrázek 15) lze získat jednofázový přímý měnič

kmitočtu. Skladbou tří řízených čtyřkvadrantových usměrňovačů získáme cyklokonvertor viz Obrázek 15.

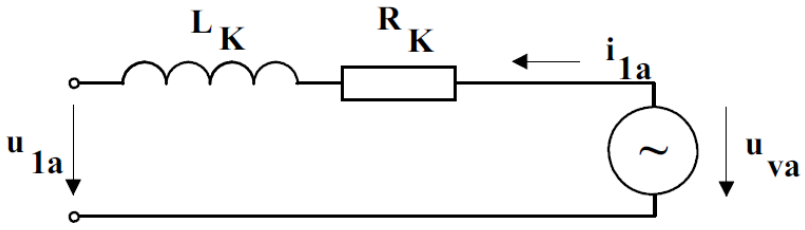


Obrázek 15 Schéma cyklokonvertoru [1]

Tvar křivky výstupního napětí je určen řídícím napětím cyklokonvertoru, které může být obdélníkové, sinusové, lichoběžníkové.

Rovnice popisující výstupní napětí cyklokonvertoru (1):

$$u_{1a} = u_{va} - R_K i_{1a} - \frac{L_K di_{1a}}{dt} \tag{1}$$



Obrázek 16 Náhradní schéma jedné fáze výkonové části cyklokonvertoru [1]

4.4 Uživatelské rozhraní

Tato kapitola popisuje uživatelské rozhraní a jeho části. Prostředkem pro vývoj uživatelského rozhraní bylo vývojové prostředí firmy National Instruments LabView 10.

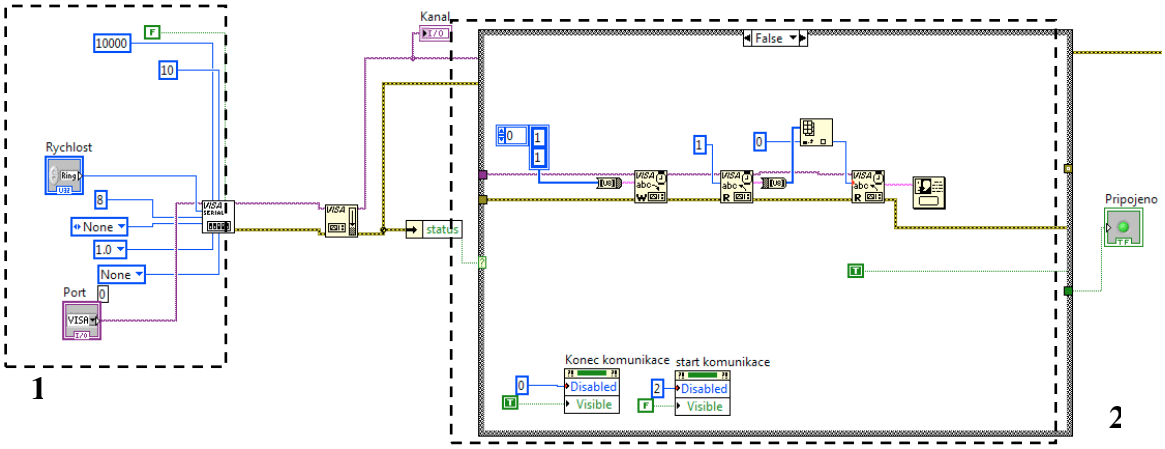
Komunikace probíhá po sériové lince RS232, rychlostí 56 000bps. Byly vytvořeny dvě verze uživatelského rozhraní (4.5.1) jedna pro U-f řízení synchronního stroje a druhá pro vektorovou regulaci (4.5.3).

Následující kapitoly popisují jednotlivé bloky uživatelského rozhraní.

4.4.1 Programový blok pro inicializaci komunikace s řídicím systémem

Tento blok zajišťuje inicializaci komunikace s řídícím systémem (4.2). Popis bloku:

1. Inicializace sériového portu (ovladače VISA)
2. Inicializace komunikace s řídícím systémem (znak „0x0001“)



Obrázek 17 Programový blok - inicializace komunikace s řídícím systémem

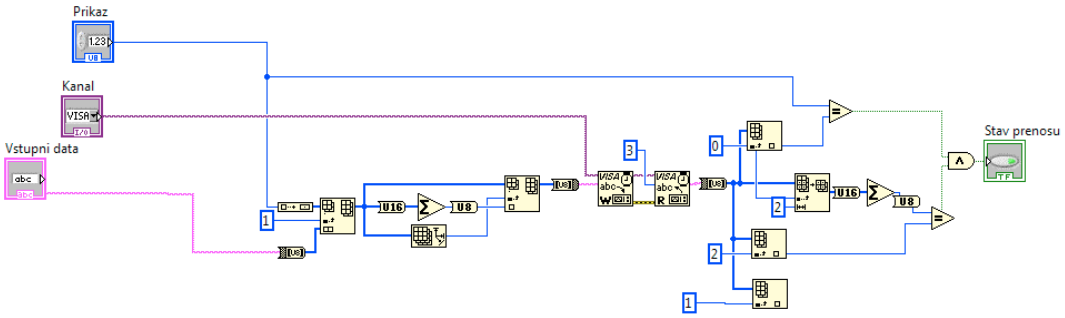
4.4.2 Programový blok pro odesílání dat do řídicího systému

Tento programový blok slouží k odesílání dat do řídícího systému po sériové linnce. Blok pro odeslání dat obsahuje také blok pro vytvoření kontrolního součtu CRC (2).

$$CRC = (l + 1) + \sum_{i=0}^{l} POLE[i] \tag{2}$$

$l \ldots délka\ pole\ s\ daty\ k\ odeslání$

Pomocí tohoto bloku se například posílá u U-f regulace požadovaná frekvence a u vektorové regulace požadovaná rychlost atd.

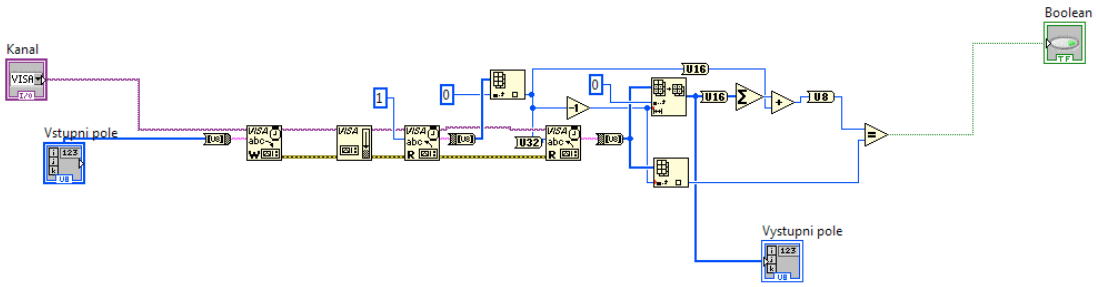


Obrázek 18 Programový blok pro odesíláni dat

4.4.3 Programový blok pro příjem dat od řídicího systému

Tento programový blok pro příjem dat z řídícho systému funguje principiálně shodně s blokem pro odesílání dat.

Tímto blokem se čtou z řídicího systému hodnoty polohy rotoru, budícího proudu rotoru. Tímto blokem je možno také posílat další data z vnitřní regulační struktury.



Obrázek 19 Programový blok pro příjem dat

4.4.4 Hlavní programový blok

Obrázek hlavního programového bloku se nachází v přílohách (**Chyba! Nenalezen zdroj odkazů.**) a je rozdělen na následující bloky:

1. Mimo hlavní smyčku se nachází výchozí nastavení hodnot regulátorů a objekty ovládacích prvků.
2. Hlavní programová smyčka, která zajišťuje vypsání přečtených hodnot z řídícího systému do prvků v uživatelském rozhraní a posílání hodnot z ovládacích prvků do řídicího systému. Všechny tyto objekty jsou zapouzdřeny v událostní struktuře, která reaguje na změnu parametrů, které jsou odesílány pravidelně každých 100ms do řídicího systému. S časovou prodlevou jsou data také čtena z řídícího systému do uživatelského rozhraní.

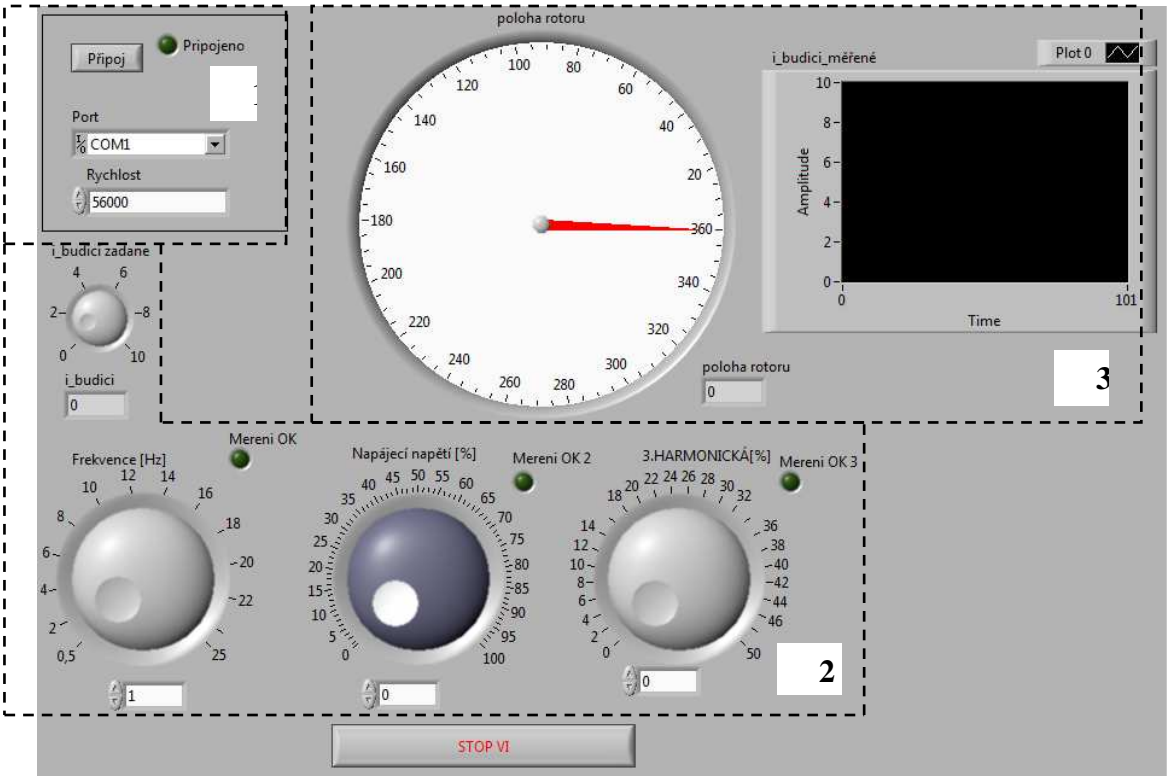
4.4.5 Grafická část uživatelského rozhraní

Tato kapitola popisuje uživatelské rozhraní vytvořené ve vývojovém prostředí labview10, které slouží k ovládání řídícího systému a tím i k ovládání motoru.

4.4.5.1 Grafická část uživatelského rozhraní pro regulaci typu U-f

Uživatelské rozhraní se skládá z následujících podskupin uživatelských a funkčních bloků:

1. Ovládací prvky pro připojení k řídicímu systému, pokud nedojde k úspěšnému navázání komunikace dojde k vypsání chyby. Pokud dojde k úspěšnému navázání komunikace je vypsána zpráva „SystemTMS320F28335, program komunikace v 0.1“
2. Uživatelské rozhraní umožňuje nezávisle měnit frekvenci (výchozí hodnota frekvence je 1Hz), napájecí napětí (0-100% rozsahu napětí regulačního trafa, výchozí hodnota je 0%) a dále umožňuje přidat 3. harmonickou pro ukázku sledování projevu na výstupním proudu (až do úrovně 50% z aktuální hodnoty napájecího napětí, výchozí hodnota je 0%). Dále je možné řídit velikost budicího proudu i_b (0-10A, výchozí hodnota je 0A)
3. Tato část umožňuje sledovat aktuální polohu rotoru a průběh budícího proudu i_b

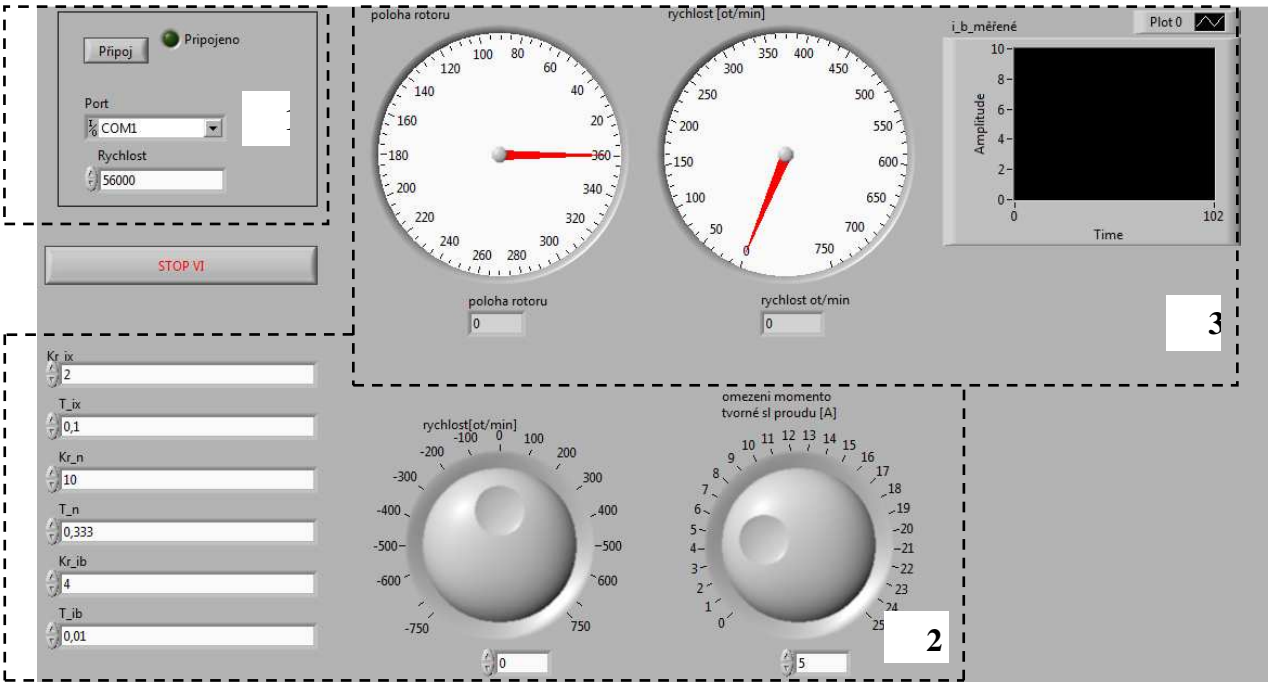


Obrázek 20 Ovládací panel k U-f regulaci

4.4.5.2 Grafická část uživatelského rozhraní pro vektorovou regulaci

Uživatelské rozhraní se skládá z následujících podskupin uživatelských a funkčních bloků:

1. Ovládací prvky pro připojení k řídicímu systému, pokud nedojde k úspěšnému navázání komunikace dojde k vypsání chyby. Pokud dojde k úspěšnému navázání komunikace je vypsána zpráva „SystemTMS320F28335, program komunikace v 0.1“
2. Uživatelské rozhraní umožňuje nezávisle měnit směr otáčení (jak ve směru hodinových ručiček tak i naopak) a rychlost. Druhý otočný ovládací prvek umožňuje měnit omezení momentotvorné složky proudu. Dále je možno měnit parametry PI regulátoru (otáček, momentotvorné a budící složky proudu).
3. Tato část umožňuje sledovat aktuální polohu rotoru, rychlost motoru a průběh budícího proudu i_b.



Obrázek 21 Ovládací panel k U-f regulaci

4.5 Programová část řídícího systému

Software je psán v programovacím jazyce C, který umožňuje přehlednější tvorbu zdrojového kódu. Software pro řídicí systém je vyvíjen ve vývojovém prostředí Code Composer Studio, které je nadstavbou

4.5.1 Hlavní část programu

Zdrojový kód hlavní části programu je uveden v příloze (**Chyba! Nenalezen zdroj odkazů.**). V této části kódu dochází k nastavení vektoru přerušení, dále dochází k inicializaci časovače CPU-TIMER0. Poté je inicializován kvadraturní modul pro čidlo otáček, komunikace SPI, SCI, AD převodník, následovaný sekcí nastavování výchozích parametrů regulátorů. Následuje hlavní smyčka programu, ve které se přijímají a zpracovávají řídící zprávy s uživatelského rozhraní (4.4).

4.5.2 Komunikace s uživatelským rozhraním v Labview

Úsek kódu sloužící k rozkódování typu příchozí zprávy z uživatelského prostředí. Tato část kódu je programována jako stavový automat s parametrem identifikačního znaku – proměnná „znak“.

```
switch (znak)
     {
            case 0x0001://uvitani
                  while(SciaRegs.SCIFFRX.bit.RXFFST != 1){}
                  if(znak = SciaRegs.SCIRXBUF.all)
                  {
                        while(SciaRegs.SCIFFTX.bit.TXFFST == 16){}
                        SciaRegs.SCITXBUF = 44;
                        SciaPrintString("System TMS320F28335\nProgram
                                          komunikace v 0.1");
                   }
                   break;
     .
     .
     .
```

4.5.3 Popis výsledné vektorové regulace v orientovaných souřadnicích (x,y)

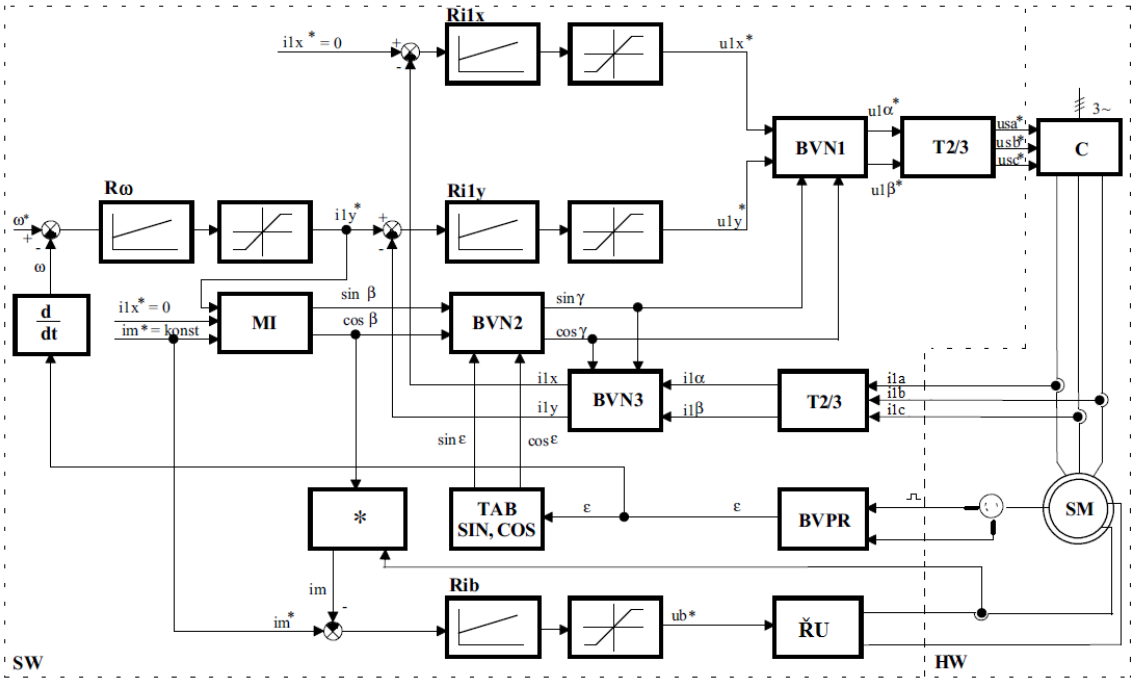
Všechny výpočty spojené s vektorovou regulací probíhají v přerušení od časovače CPU-TIMER0 (4.2.3). Pro výpočty otáčkového regulátoru je nastavena interval regulace 5ms. Pro regulátory (budícího proudu, momentotvorného proudu a veškeré matematické operace spojené s vektorovou regulací) mají nastaven interval 500µs. Přerušení od časovače spouští AD převodníky.

```
interrupt void cpu_timer0_isr(void)

{
     pos = EQep1Regs.QPOSCNT;
     EPS = ((float32)pos)*0.00306796; //přepočet polohy na radiány
     sin_eps = sin(EPS);
     cos_eps = cos(EPS);
     int_cnt++;
     if(int_cnt == SPEED_AVG_SAMPLES) //smyčka výpočtu rychlosti a
regulace otáček
     {
           d_pos = pos - old_pos;
           if(d_pos > 2047)
           {
                 d_pos = d_pos- 4095;
           }
           else if(d_pos < -2047)
           {
                 d_pos = d_pos+ 4095;
           }

           w_m = ((0.3067961) * ((float32)d_pos));     //vypocet rychlosti
           w_m_osc = w_m * 0.004;   //norma 250ot/min je 10Vna výstupu
                                      DA(DA má zes 10 )
           iq_z = i_model.iy_z;//Regulator otacek pro vekt v rotorových
                                souøadnicích
                              //Vychozi nastaveni
           int_cnt=0;
           old_pos = pos;
     }
     /*********************************************************************
     řízení v orientovaných souřadnicích
     *********************************************************************
*/
     ProudModel(&i_model, &im_sinb, &im_cosb); //MI - Vypocet parametru
                                          proudoveho modelu -> sinB, cosB
     VektRot1(im_cosb, im_sinb, &cos_gama, &sin_gama, sin_eps,
               cos_eps);           //BVN2 - Vektorove natoceni 2.
     T3_2E(i_a, i_b, i_c, &i_alfa, &i_beta);//T3/2 - Transformace 3/2
                                          vystupniho proudu
                                          cyklokonvertoru na
                                          statorove souradnice alfa a
                                          beta
     VektRot2(i_alfa, i_beta, &i_xsk, &i_ysk, sin_gama, cos_gama);
     //BVN3 - Blok vektoroveho natoceni 3. statorovych souradnic alfa, beta
       do souradnic x,y
     u_yR = Reg_PI(i_model.iy_z, i_ysk, &iyreg);//vystup z reg.proudu
                                                budici slozky proudu
     u_xR = Reg_PI(i_xz, i_xsk, &ixreg);//vystup z reg.proudu
                                          momentotvorne slozky proud
     VektRot1(u_xR, u_yR, &u_alfa, &u_beta, sin_gama, cos_gama);
                                                      //BVN1
     T2_3(u_alfa, u_beta, &ua, &ub, &uc); //T2/3vystupni ridici napeti
```

```
im_sk = (i_budsk * im_cosb);
u_bud = Reg_PI_om(i_budz,im_sk,&regib);//vystup z reg proudu
```



Obrázek 22 Blokové schéma vektorové regulace v orientovaných souřadnicích [1]

Z hodnot žádaných a skutečných otáček generuje PI regulátor otáček žádaný momentotvorný proud.

Do bloku proudového modelu, který je popsán rovnicemi[1]:

$$sin\beta = \frac{i1y^*}{\sqrt{\left(K_p * i1y^*\right)^2 + im^2}}$$

$$cos\beta = \frac{i_m^*}{\sqrt{\left(K_p * i1y^*\right)^2 + i_m^{*2}}}$$

vstupují hodnoty žádaných proudu i_y, i_x žádaného magnetizačního proudu i_m, který je konstantní. Z proudového modelu dostaneme hodnoty sinβ a cosβ, tyto signály vstupují do bloku vektorového natočení 2 (BVN2). Do bloku BVN2 vstupují také přepočtené hodnoty otáčkového čidla sinε a cosε. Výstupem bloku BVN2 je sinγ a cosγ tyto hodnoty vstupují do bloku BVN3 přepočtených statorových proudů i_a, i_b, i_c pomocí bloku transformace T3/2 (čteme zprava doleva) na iα, iβ, které následně vstupují do bloku BVN3. Výstupem jsou hodnoty momentotvorného proudu i_y a magnetizačního proudu i_x. Tyto hodnoty vstupují do PI regulátorů proudů i_x, i_y. Výsledkem jsou hodnoty u_x, u_y, které vstupují do bloku BVN1, který

natáčí hodnoty u_x, u_y do statorového souřadnicového systému uα, uβ, poté vstupují do bloku transformace 2/3 z nějž vystupují řídící veličiny pro cyklokonvertor.

Hodnota cosβ se násobí z hodnotou skutečného budícího proudu i_b, výsledkem je skutečné i_m (magnetizační proud), tato složka vstupuje do PI regulátoru budícího proudu, výsledkem řídící napětí pro řízený usměrňovač.

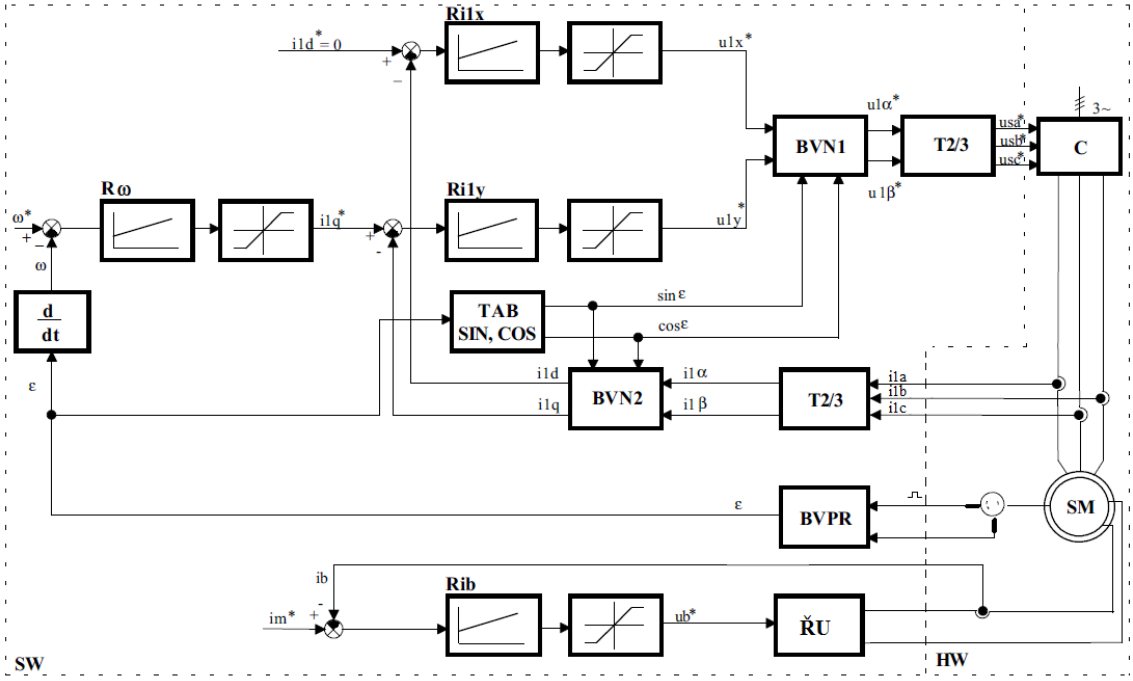
Pro rozběh motoru je nutné znát skutečnou polohu rotoru vůči statoru. Tuto hodnotu získáme tak, že před spuštěním výkonové části otočíme manuálně rotorem dokud nedojde k vynulování, nulovacím pulsem čidla otáček.

4.5.4 Popis výsledné vektorové regulace v rotorových souřadnicích (DQ)

Zdrojový kód pro vektorovou regulaci v rotorových souřadnicích je shodný se zdrojovým kódem vektorové regulace v orientovaných souřadnicích (4.5.3) až na následující část kódu.

```
T3_2E(i_a, i_b, i_c, &i_alfa, &i_beta);
VektRot2(i_alfa, i_beta, &id, &iq, sin_eps, cos_eps);//BVN2 - Vektorove
                                                      natoceni 2.
u_yR = Reg_PI(iq_z, iq, &iyreg);//vystup z reg.proudu budici slozky
proudu
u_xR = Reg_PI(id_z, id, &ixreg);//vystup z reg.proudu momentotvorne
slozky
                                  proudu
VektRot1(u_xR, u_yR, &u_alfa, &u_beta, sin_eps, cos_eps);//BVN1
T2_3(u_alfa, u_beta, &ua, &ub, &uc);      //T2/3vystupni ridici napeti

u_bud = Reg_PI_om(i_budz,i_budsk,&regib);
```



Obrázek 23 Blokové schéma vektorové regulace v rotorových souřadnicích

Blokové schéma na obrázku (Obrázek 23) je shodné s blokovým schématem vektorové regulace v orientovaných souřadnicích. V tomto případě nevyužíváme proudového modelu, ale přímo hodnoty sinε cosε posíláme do bloku vektorového natočení BVN2 do nějž vstupní také proudy iα, iβ, které se natáčí do rotorového systému DQ.

4.5.5 Popis výsledné U-f regulace

Při U-f regulaci není zavedena zpětná vazba statorových veličin. V proměnné uhel a uhel_b se vytváří hodnota v radiánech pro funkci sin respektive cos, která následně generuje harmonickou funkci. Tyto dvě harmonické funkce transformací 2/3 generují výsledné řídící signály pro stator.

Následuje regulace budícího proudu, kterou zadáváme uživatelském rozhraní. Tento budící proud může být i nulový, v tomto případě se využívá zbytkové remanentní magnetizace rotoru.

```
interrupt void cpu_timer0_isr(void)
{
   uhel += ((2*PI)/(2000/frekvence));
   if(uhel > 2*PI)
   {
     uhel = 0;
   }
   uhel_b = uhel + (PI/2);
   if(uhel_b > 2*PI)
   {
     uhel_b -= 2*PI;
   }
```

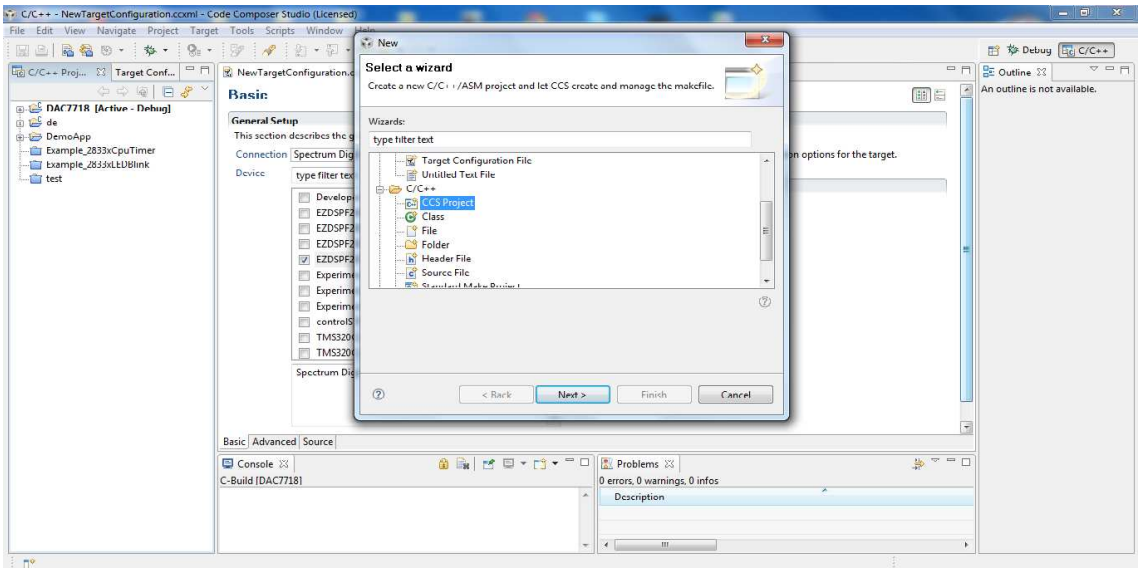
```
    u_bud = Reg_PI_om(i_budz,i_budsk,&regib);
    T2_3(sin(uhel), sin(uhel_b),&a,&b,&c);
    ua=(prvnih/100)*((a)+((sin(uhel*3))*(tretih*0.01)));
    ub=(prvnih/100)*((b)+((sin(uhel*3))*(tretih*0.01)));
    uc=(prvnih/100)*((c)+((sin(uhel*3))*(tretih*0.01)));
    Send_DAC7718(VOUT_7,((int16)(32767 * ua)));
    Send_DAC7718(VOUT_6,((int16)(32767 * ub)));
    Send_DAC7718(VOUT_5,((int16)(32767 * uc)));
    Send_DAC7718(VOUT_4,((int16)(32767 * u_bud)));
    Ldac();
    GpioDataRegs.GPBTOGGLE.bit.GPIO32 = 1;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;//vymazani priznaku preruseni
    AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 1;
}
```

4.5.6 Založení nového projektu ve vývojovém prostředí Code Composer

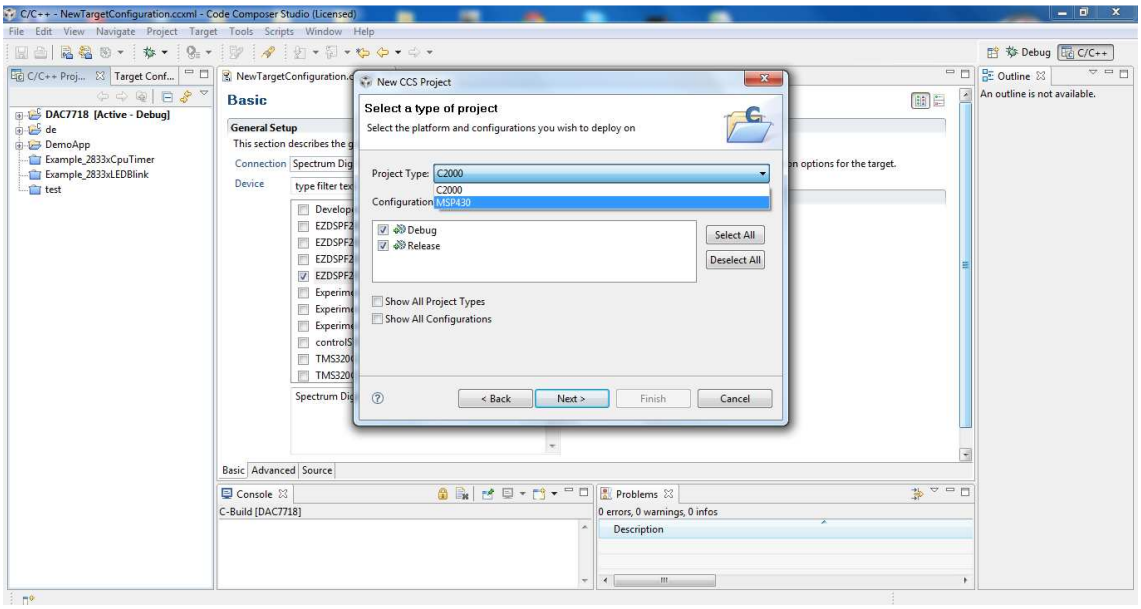
Tato kapitola popisuje založení nového projektu ve vývojovém prostředí Code Composer pro verzi procesoru TMS320F28335. Založení nového projektu je prováděno skrze průvodce vytvořením nového projektu což velice usnadňuje proces vytvoření projektu.

Postup založení nového projektu:

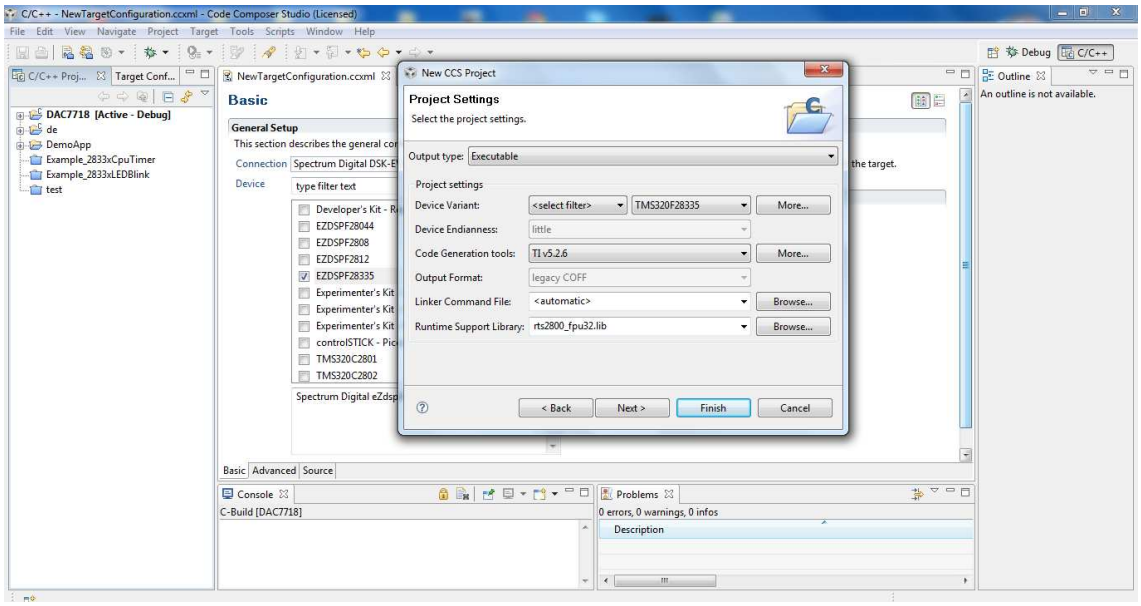
1. Pro vytvoření nového projektu, vybereme v nabídce (File → New → CSS Project). Objeví se dialogové okno, které lze vidět na obrázku (Obrázek 24). V tomto dialogovém okně necháme výchozí zvolenou hodnotu a klepneme na tlačítko Next.
2. V dalším dialogovém okně (Obrázek 25) dochází k výběru rodiny procesorů. Pro procesor TMS320F28335 vybereme možnost C2000 a klepneme na tlačítko next
3. V dalším dialogovém okně (Obrázek 26) dochází k výběru konkrétního modelu procesoru a k verzi používaného kompilátoru a linkeru. Dále je zde možnost vybrat knihovny, které chceme pro Nás projekt použít (například knihovny pro výpočty v plovoucí čárce). Po výběru klepneme na tlačítko next
4. V posledním dialogovém okně (Obrázek 28) dochází k výběru předlohy nového projektu, kde vývojové prostředí může vytvořit prázdný projekt nebo může podle šablony vytvořit ukázkový projekt typu „Hello World“. Po výběru šablony projektu klepneme na tlačítko finish.
5. V posledním kroku (Obrázek 29) můžeme začít psát samotný program a po stisku kombinace kláves CTRL + B jej můžeme zkompilovat nebo případně pomocí tlačítka „Debug“ v menu Target.



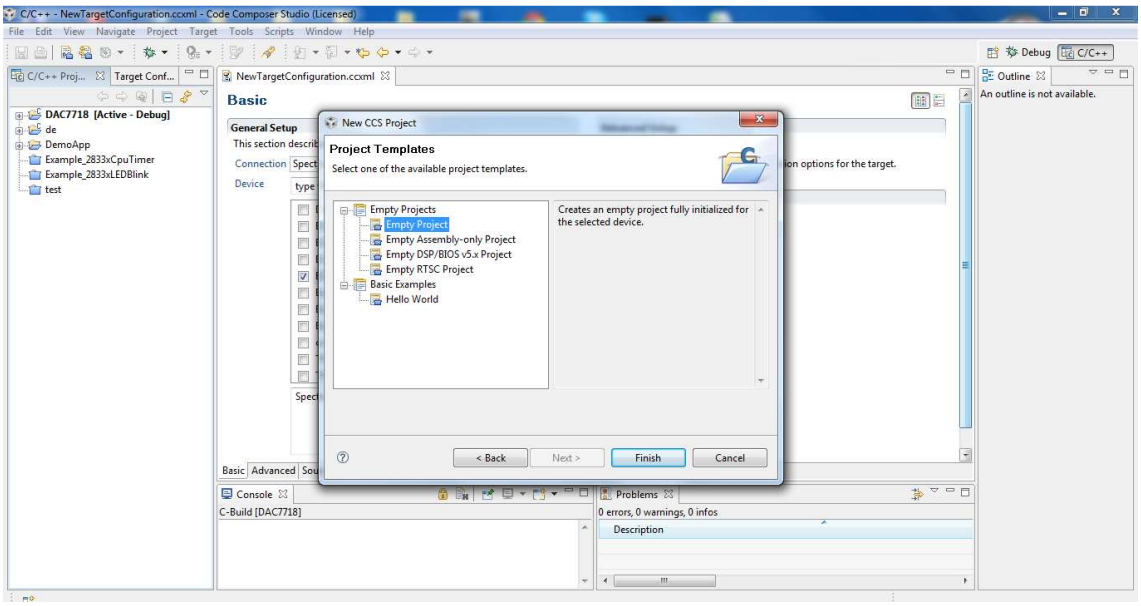
Obrázek 24 První krok při založení nového projektu



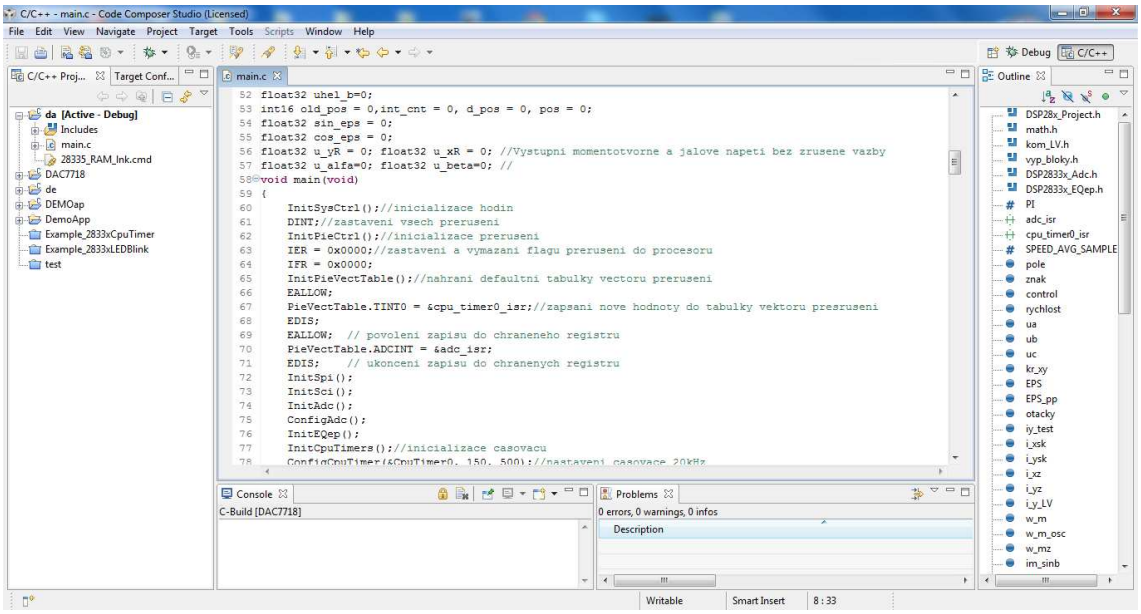
Obrázek 25 Volba rodiny procesoru



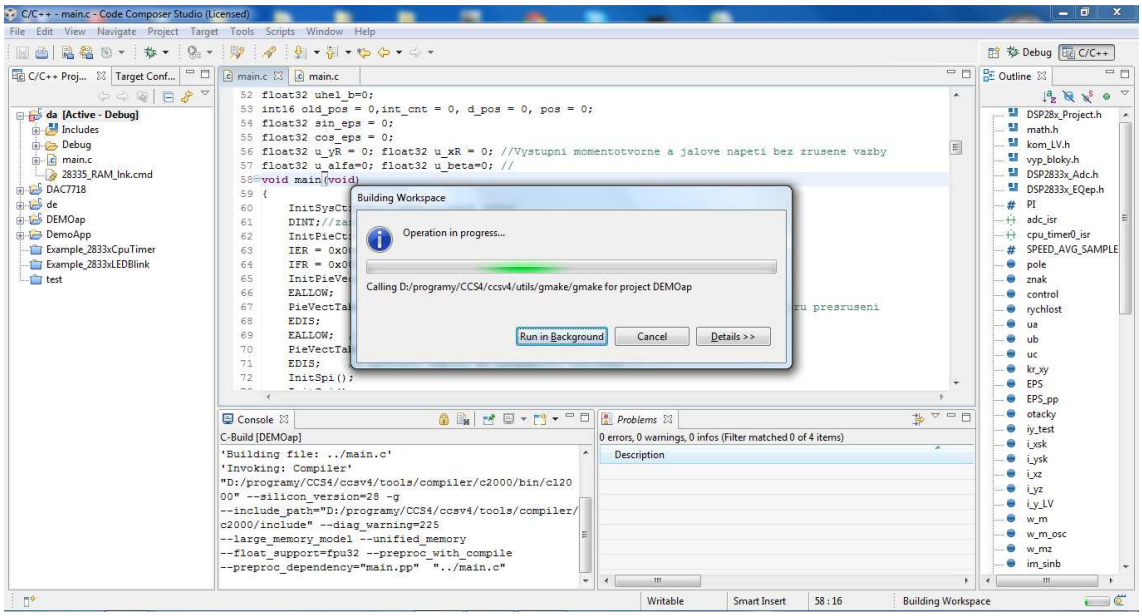
Obrázek 26 Volba procesoru a verze kompilátoru, linkeru, použitých knihoven



Obrázek 27 Volba předlohy projektu



Obrázek 28 Výchozí okno programového prostředí s aktivním projektem



Obrázek 29 Kompilace projektu

4.6 Návrh a realizace desky plošných spojů

Tato kapitola popisuje návrh desky plošných spojů pomocí programu Eagle od firmy CADsoft. Deska byla navrhována jako oboustranná deska bez prokovu. Výsledek lze vidět na obrázku.

4.6.1 Schéma

Schémata desek plošných spojů jsou uvedeny v přílohách (**Chyba! Nenalezen zdroj odkazů.**, **Chyba! Nenalezen zdroj odkazů.**, **Chyba! Nenalezen zdroj odkazů.**, **Chyba!**

Nenalezen zdroj odkazů.), které zobrazují schémata (desky rozhraní k vývojovému kitu eZdsp, galvanického oddělení pro čídlo otáček a stabilizační část napájení, externího DA převodníku a buzení pro moduly PWM, převodníků napěťové úrovně pro AD převodník)

- Schéma rozhraní k vývojovému kitu eZdsp
  - Toto schéma obsahuje propojení vstupně výstupních portů z vývojové sady eZdsp do rozšiřující desky řídícího systému
  - Dále toto schéma obsahuje ochranné schotkyho diody pro vstup AD převodníku v procesoru

- Schéma galvanického oddělení čidla otáček a stabilizační část napájení
  - Toto schéma obsahuje stabilizátory napětí 3V3 a 1V5. Výstup stabilizátoru 1V5 slouží jako referenční napětí pro převodníky úrovní (+-10V → 0 – 3V) Výstup ze stabilizátoru 3V3 slouží k napájení výstupní části izolačního členu
  - Dále toto schéma obsahuje izolační člen (ISO7240CF), který galvanicky odděluje napětí mezi čidlem otáček a rozšiřující deskou
  - Toto schéma obsahuje napájecí Mini-DIN konektor (4.6.3)
  - Dále obsahuje DC-DC měnič 5V → 5V, který slouží pro napájení čidla otáček

- Schéma zapojení externího DA převodníku a buzení pro moduly PWM
  - Toto schéma obsahuje zapojení externího DA převodníku (DAC7718)
  - Dále toto schéma obsahuje budící obvody pro PWM výstupy procesoru

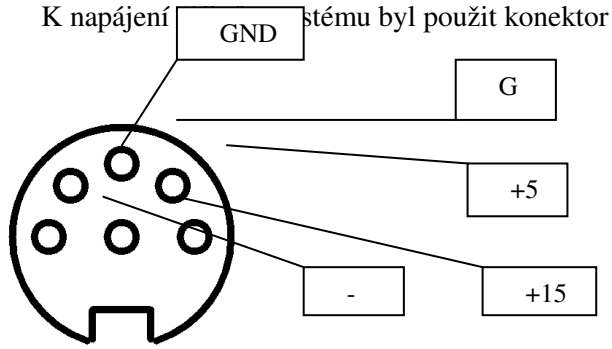
- Schéma převodníků napěťové úrovně pro AD převodník
  - Toto schéma obsahuje převodníky napěťových úrovní (+-10V → 0 – 3V), tvořené sítí operačních zesilovačů OPA4227 firmy Burn&Brown

4.6.2 Deska plošných spojů

Deska plošných spojů (**Chyba! Nenalezen zdroj odkazů.**). Jedná se o oboustrannou desku bez prokovů o rozměrech 145x165mm.

4.6.3 Napájecí konektor

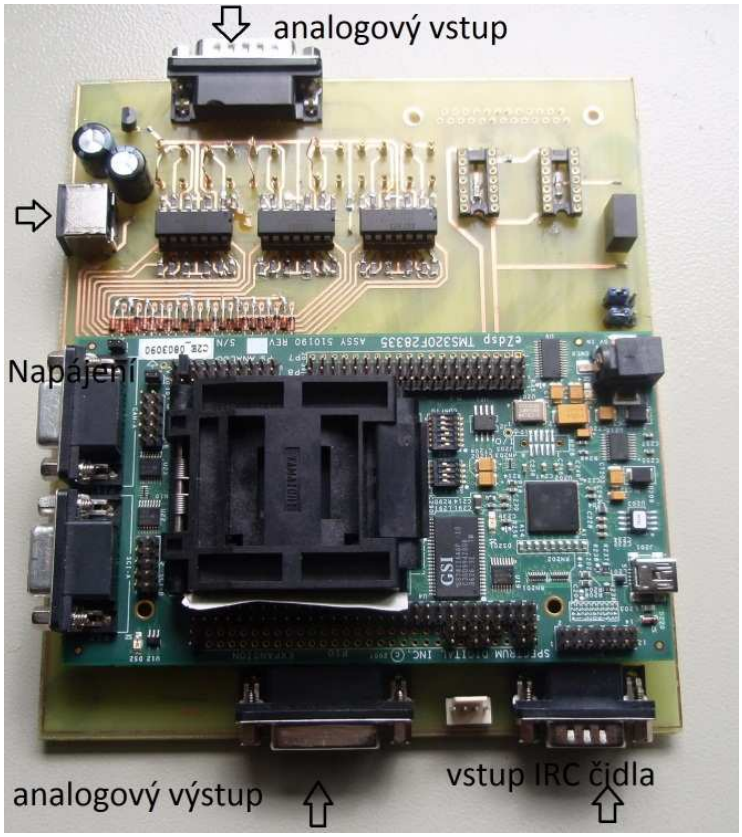
K napájení [illegible]stému byl použit konektor Mini-DIN (Obrázek 30)



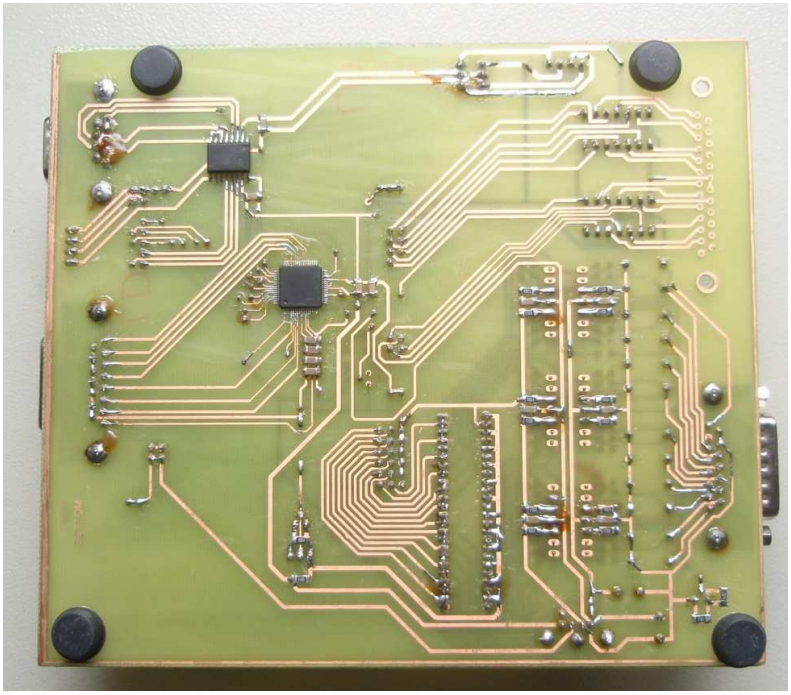
Obrázek 30 Mini-DIN

4.6.4 Výsledná realizace desky rozšiřujícího rozhraní

Pohled na vrchní (Obrázek 31) a spodní (Obrázek 32) část desky rozšiřujícího rozhraní s nasazenou vývojovou deskou eZdsp.



Obrázek 31 pohled na vrchní část desky



Obrázek 32 pohled na spodní část desky

Obrázek 33 testovaný motor s cyklokonvektorem

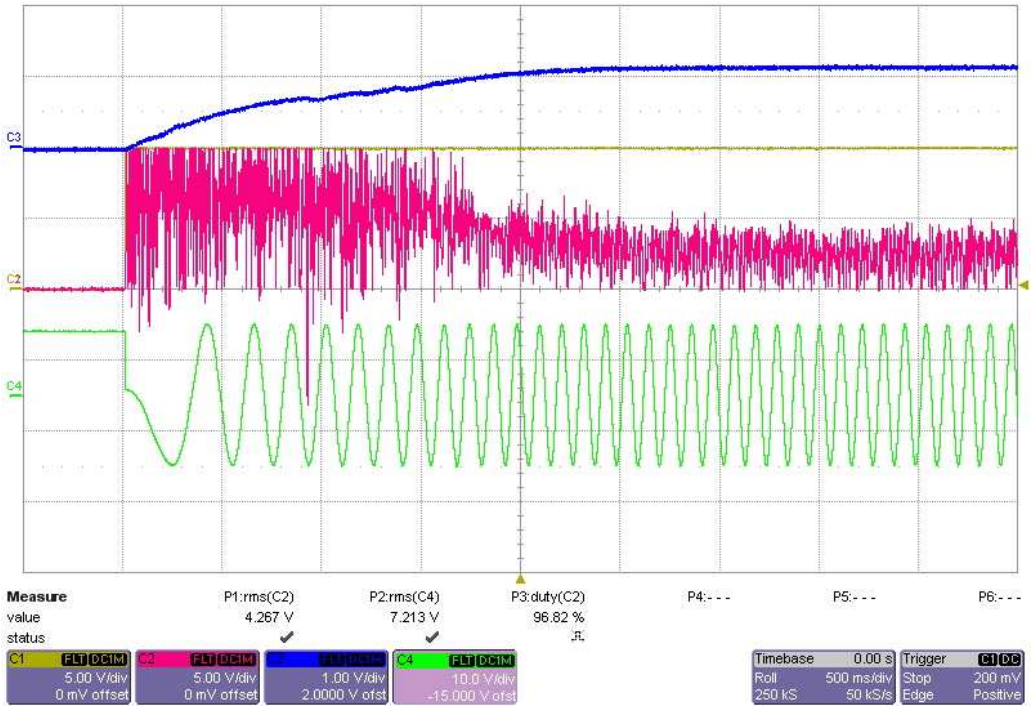
5 Naměřené výsledky

Tato kapitola zobrazuje a popisuje průběhy naměřené na již realizovaném řešení celého systému.

5.1 Průběh rozběhu motoru z nulových otáček, vektorová regulace v orientovaném systému x,y

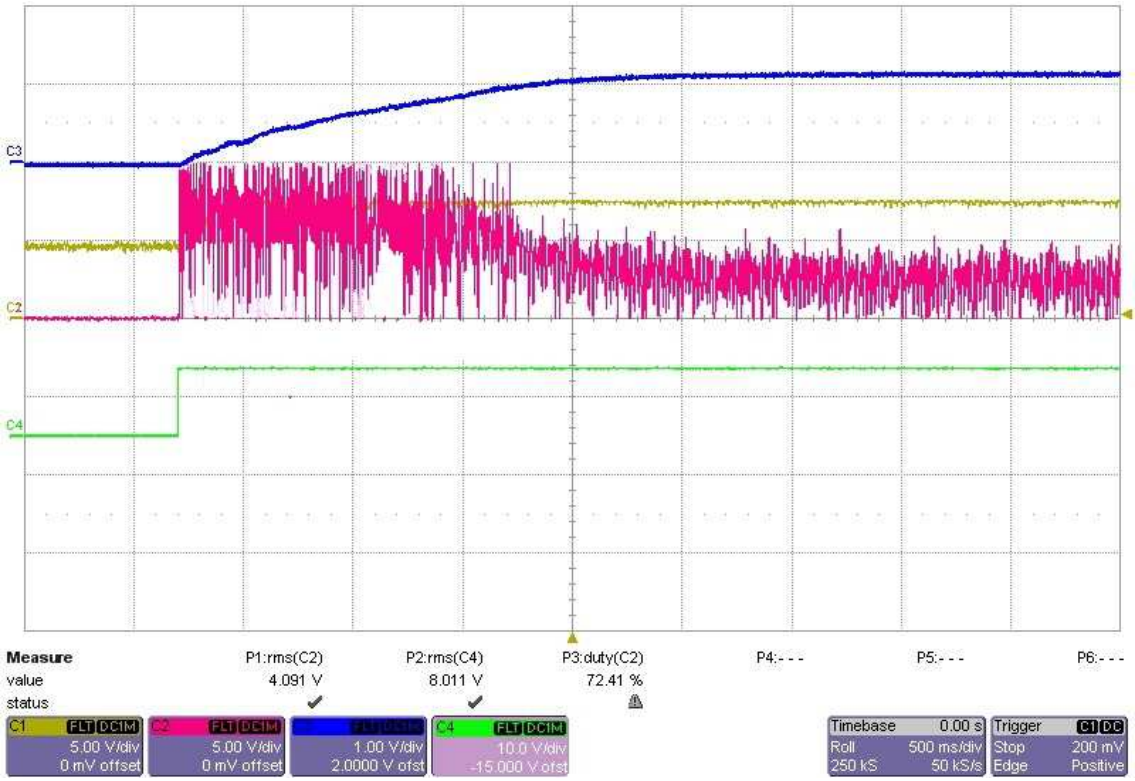
Průběh na obrázku (Obrázek 35) zobrazuje rozběh synchronního motoru s budícím vinutím v aplikaci vektorové regulace v orientovaném systému. Měřené veličiny:

Kanál osciloskopu	Barva průběhu	Veličina	Norma
1	Žlutá	i_{y_zadana}	10A / dílek
2	Růžová	$i_{y\_skutecna}$	10A / dílek
3	Modrá	ω_m	250 otáček/dílek
4	Zelená	$\sin\gamma$	$\pi/2$ na 1. dílek



Obrázek 34 Průběh rozběhu motoru z nulových otáček na proudové omezení 8A

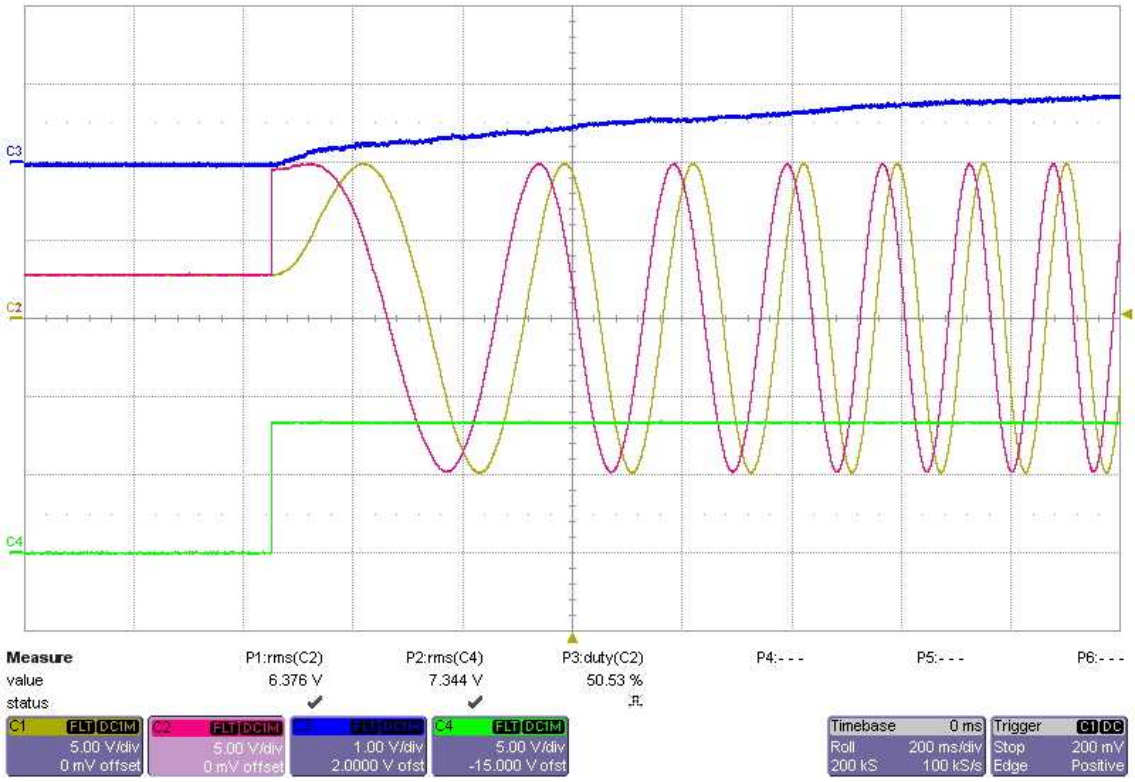
Kanál osciloskopu	Barva průběhu	Veličina	Norma
1	Žlutá	$i_{b\_skutecna}$	10A / dílek
2	Růžová	$i_{y\_skutecna}$	10A / dílek
3	Modrá	ω_m	250 otáček/dílek
4	Zelená	sinβ	π/2 na 1. dílek



Obrázek 35 Průběh rozběhu motoru z nulových otáček na proudové omezení 8A

Na tomto obrázku (Obrázek 36) pozorujeme fázový posuv mezi sinε a sinγ, který vytváří v kombinaci sinβ

Kanál osciloskopu	Barva průběhu	Veličina	Norma
1	Žlutá	Sinε	π/2 na 1. dílek
2	Růžová	sinγ	π/2 na 1. dílek
3	Modrá	ω_m	250 otáček/dílek
4	Zelená	sinβ	π/2 na 1. dílek

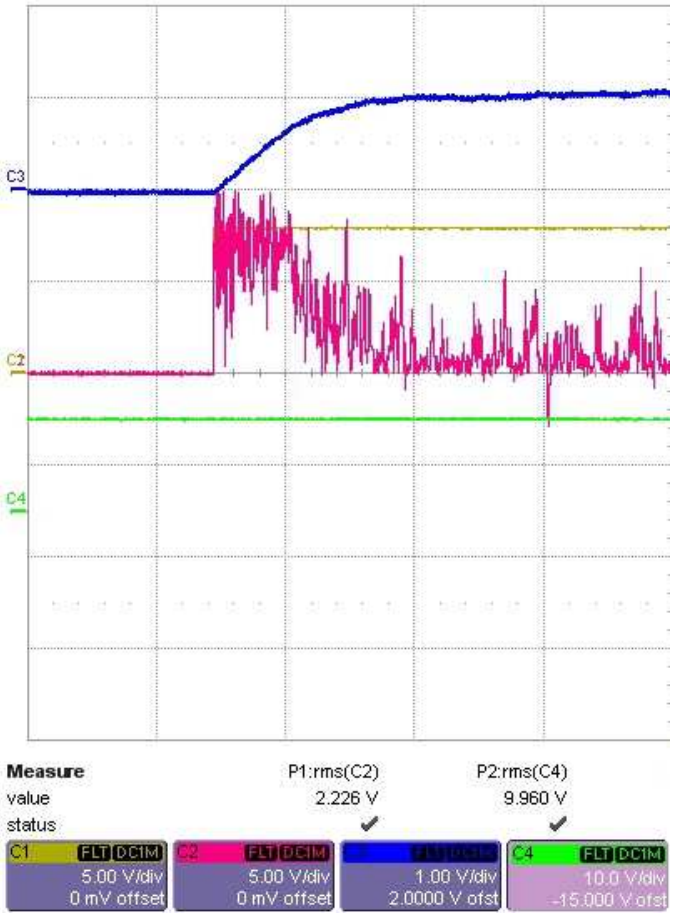


Obrázek 36 Průběh rozběhu motoru z nulových otáček na proudové omezení 8A

5.2 Průběh rozběhu motoru z nulových otáček na proudové omezení 8A v rotorových souřadnicích DQ

Průběh na obrázku (Obrázek 37) zobrazuje rozběh synchronního motoru s budícím vinutím v aplikaci vektorové regulace v rotorovém systému DQ. Měřené veličiny:

Kanál osciloskopu	Barva průběhu	Veličina	Norma
1	Žlutá	i_{q_zadana}	10A / dílek
2	Růžová	$i_{q\_skutecna}$	10A / dílek
3	Modrá	ω_m	250 otáček/dílek
4	Zelená	sinε	π/2 na 1. dílek



Obrázek 37 Průběh rozběhu motoru z nulových otáček na proudové omezení 8A v rotorových souřadnicích DQ

6 Závěr

Úkolem této práce bylo vytvořit řídicí systém s aplikovaným algoritmem vektorové regulace na reálném synchronním motoru s cizím buzením. Při řešení řídicího systému bylo použito moderních vývojových prostředků v podobě grafického, vývojového prostředí labview10 a programovacího jazyka C, který poskytuje vyšší úroveň programování než jazyk assembler.

Vektorová regulace nefungovala zcela správně, což pozorujeme na obrázcích v kapitole 5. což lze pozorovat na měřeném momentotvorném proudu i_y, který je rozkmitaný což je pravděpodobně způsobeno větším zašuměním signálu statorového proudu. Je tedy doporučeno vylepšit toto řešení po stránce filtrace statorových proudů, bude nutný zásah do realizované desky rozšiřujícího rozhraní. Ani po delším testování a ladění parametrů regulátoru, nedošlo k odstranění šumu ze signálu statorových proudů.

Současný stav řešení je do budoucna možno vylepšit a doplnit o vyhodnocení polohy statoru před rozběhem pomocí metody vyhodnocení počáteční polohy rotoru během počátečního buzení rotoru synchronního motoru.

Uživatelské rozhraní lze do budoucna rozšířit o ochranu proti rozběhu, dokud nedojde k nulování polohového čidla rotoru.

Realizovaná rozšiřující deska není vázaná pouze na aplikaci řízení synchronního motoru, ale lze ji například použít i pro vektorové řízení asynchronního motoru za využití modulů pulzně šířkové modulace pro generování průběhů příslušných průběhů, které umožní řídit nepřímý měnič kmitočtu. Deska obsahuje DC-DC měnič pro galvanicky oddělené napájení čidla otáček. Tento DC-DC má však maximální proud 200mA. Na desce je kvůli tomuto omezení vytvořen přepínač pro přivedení externího napájení čidla otáček.

Použitá literatura

[1] Branštetr, P. Moderní způsoby řízení střídavých strojů. VŠB – Technická univerzita Ostrava. Ostrava, 1999

[2] Chmelík, Karel. Asynchronní a synchronní elektrické stroje. VŠB – Technická univerzita Ostrava. Ostrava, 2002, 1. vydání. ISBN: 80-248-0025-X.

[3] Antošová, Marcela; Davídek, Vratislav. *Číslicová technika.* České Budějovice: Kopp, 2007, 286 s. ISBN 978-80-7232-314-2.

[4] Palacký, Petr. Skripta VŠB - TU Ostrava - *Mikroprocesorové řídicí systémy I*

[5] Texas Instruments, Incorporated. *Selecting the Right Texas Instruments Signal Switch* [online]. 18.10.2001, poslední revize 13.2.2004. Dostupné z http://focus.ti.com/lit/an/szza030/szza030.pdf

[6] Skalický, Petr. Aplikace signálovýchprocesoru: Praha, 2002, 1. vydání. ISBN 80-01-02647-7

[7] Wikipedia. Digitální signálový procesor [online]. URL< http://cs.wikipedia.org/wiki/Digit%C3%A1ln%C3%AD_sign%C3%A1lov%C3%BD_procesor>

[8] Wikipedia. Harvardská architektura URL< http://cs.wikipedia.org/wiki/Harvardsk%C3%A1_architektura

[9] Smékal, Zdeněk; Sysel, Petr. Signálové procesory. Praha, 2006. ISBN 80-86645-08-8